Programski prevodioci 1
- Vežbe IX -

+8 НБ КМ УС СЧ Софија Чоловић Павле Грковић ЈМ Јован Миљковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић КМ Ксенија Младеновић

MIKROJAVA VM

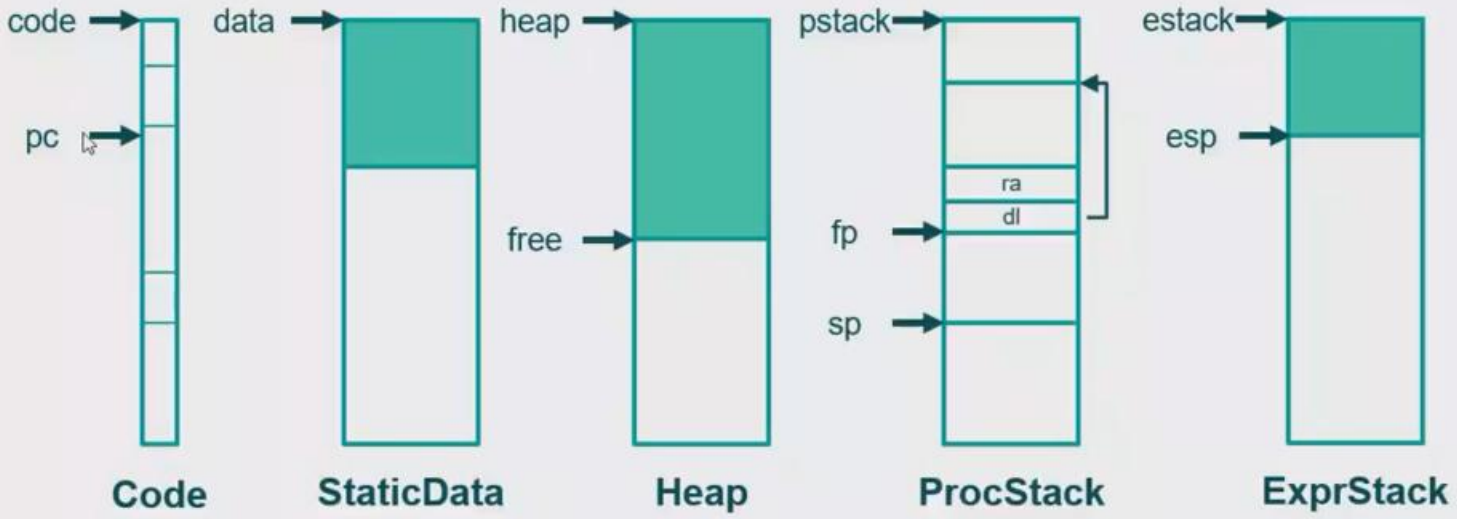
- MEMORIJSKE OBLASTI -

+11 СЧ НБ УС KZ JM KM CM PB KM
Kristijan Ziza Павле Грковић Јован Миљковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Ксенија Младеновић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:

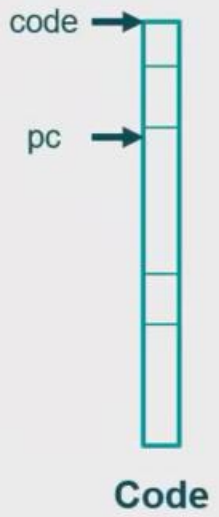


+10 МН НМ УС KZ JM КМ СМ РВ СЧ
Kristijan Žiža Павле Грковић Јован Миљковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:

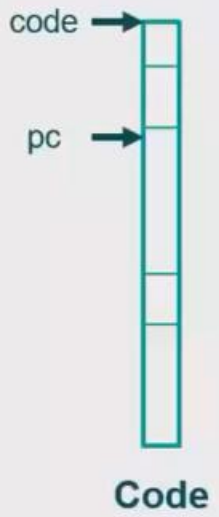


+10 МН НМ УС KZ JM КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Јован Миљковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:



Code oblast

- Ova oblast sadrži kod metoda, preciznije bajt kod programa.
- U registru **pc** se nalazi indeks instrukcije VM koja se trenutno izvršava.
- Registar **mainpc** sadrži početnu adresu metode main().
- Iz ove oblasti VM čita instrukcije i izvršava ih.
- Code oblast predstavlja niz bajtova, dok su sve ostale oblasti nizovi reči. (rec = 4B)

+11 MH HM УС KZ JM KM CM PB СЧ
Kristijan Ziza Павле Грковић Јован Миљковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:

StaticData oblast

- U ovoj oblasti se nalaze (statički ili globalni) podaci glavnog programa.
- Predstavlja niz promenljivih.
- Svaka promenljiva zauzima jednu reč (32 bita). Adrese promenljivih su indeksi pomenutog niza..

+11 МН НМ УС KZ JM КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Јован Миљковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:

Heap oblast

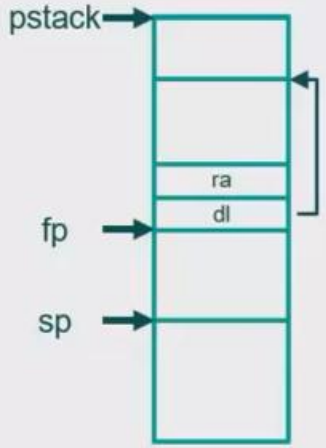
- Ova oblast sadrži dinamički alocirane objekte i nizove.
- Blokovi u heap-u se alociraju sekvencijalno. **free** pokazuje na početak slobodnog dela heap-a.
- Dinamički alocirana memorija se oslobađa samo na kraju izvršenja programa. Ne postoji sakupljanje đubreta.
- Svako polje unutar objekta zauzima jednu reč (32 bita).
- Nizovi čiji su elementi tipa char su nizovi bajtova. Njihova dužina je umnožak broja 4.
- Pokazivači su bajt ofseti u heap-u.
- Objekti tipa niza počinju “nevidljivom” rečju koja sadrži dužinu niza.

+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:



ProcStack

ProcStack oblast

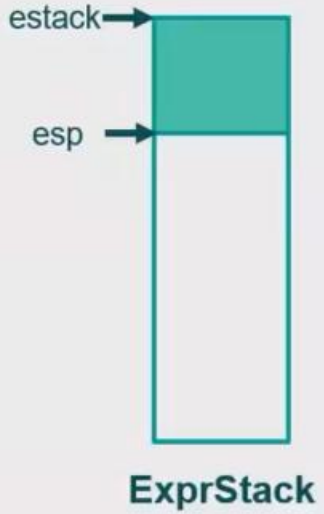
- U ovoj oblasti VM pravi aktivacione zapise pozvanih metoda.
- Svaki zapis predstavlja niz lokalnih promenljivih, pri čemu svaka zauzima jednu reč (32 bita).
- Adrese promenljivih su indeksi niza.
- **ra** (return address) je povratna adresa metode, **dl** (dynamic link) je kontrolna veza (pokazivač na aktivacioni zapis pozivaoca metode).
- Novoalocirani zapis se inicijalizuje nulama.

+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:



ExprStack oblast

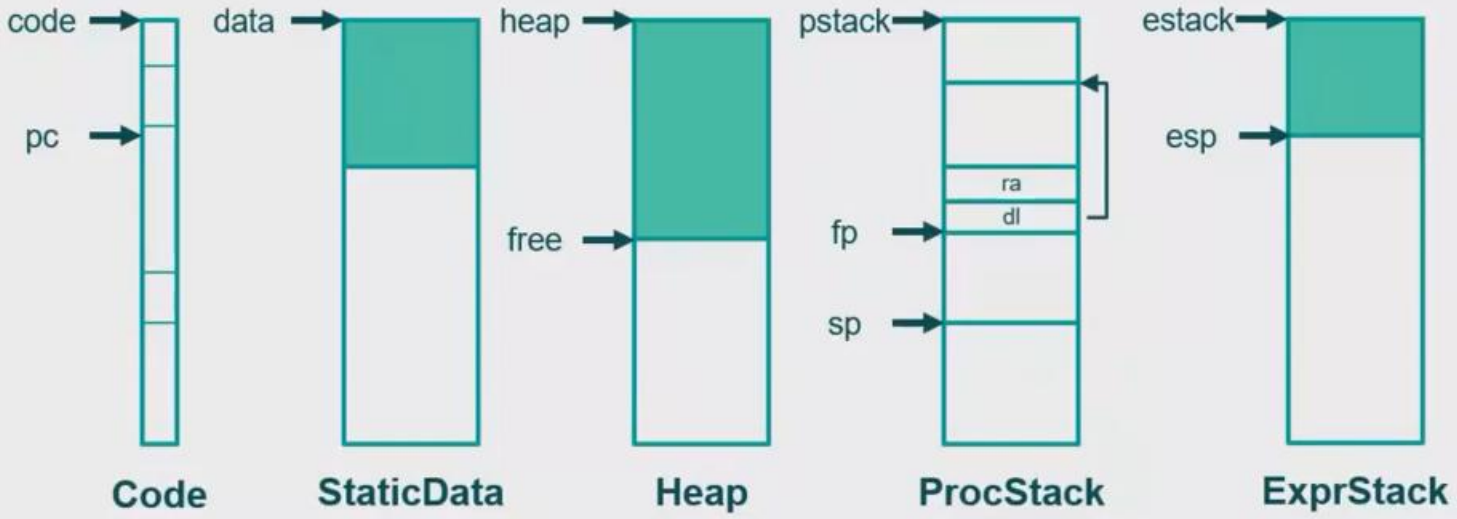
- Ova oblast se koristi za skladištenje operanada i rezultata instrukcija.
- ExprStack je prazan posle svake Mikrojava instrukcije.
- Stvarni argumenti metoda se, pre poziva metode, prosleđuju na stek izraza i kasnije uklanjaju *enter* instrukcijom pozvane metode.
- Ovaj stek izraza se takođe koristi za prosleđivanje povratne vrednosti metode njenom pozivaocu.

+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

memorijske oblasti

Mikrojava VM koristi sledeće memorijske oblasti:



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

MIKROJAVA VM
- INSTRUKCIJE -

+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

1. Instrukcije za load i store lokalnih promenljivih

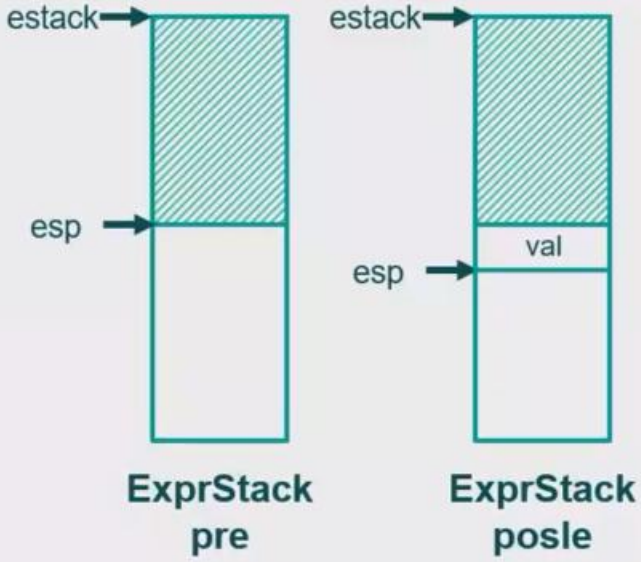
`load b` b = 1B, s = 2B, w = 4B

Koristi se za stavljanje lokalnih promenljivih na stek pri njihovom korišćenju u telu funkcija. Za promenljive sa indeksima 0, 1, 2 i 3 posebne instrukcije koje nemaju operand b: ***load_0***, ***load_1***, ***load_2*** i ***load_3***

Primer: učitavanje lokalne promenljive x na stek:

`load 0`

`(ili skraćeno) load_0`




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

1. Instrukcije za load i store lokalnih promenljivih

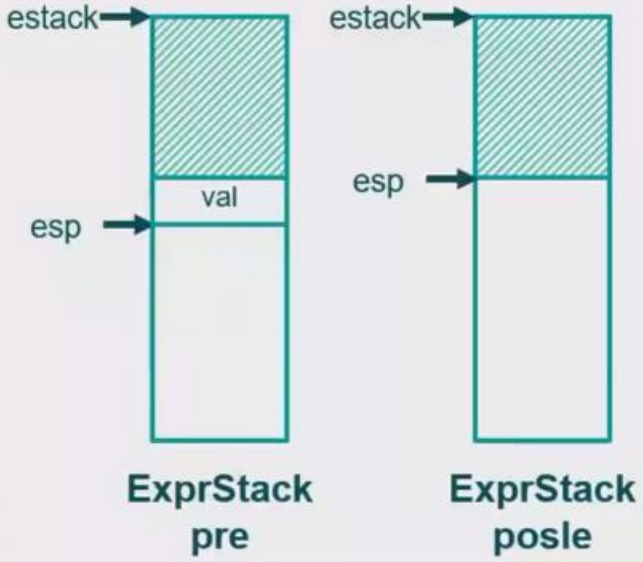
store b

Koristi se za smeštanje vrednosti sa steka u lokalne promenljive prilikom njihovog korišćenja u telu funkcija. Za promenljive sa indeksima 0, 1, 2 i 3 posebne instrukcije koje nemaju operand b: ***store_0***, ***store_1***, ***store_2*** i ***store_3***

Primer: smeštanje vrednosti u promenljivu k:

store 2

(ili skraćeno) store_2




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

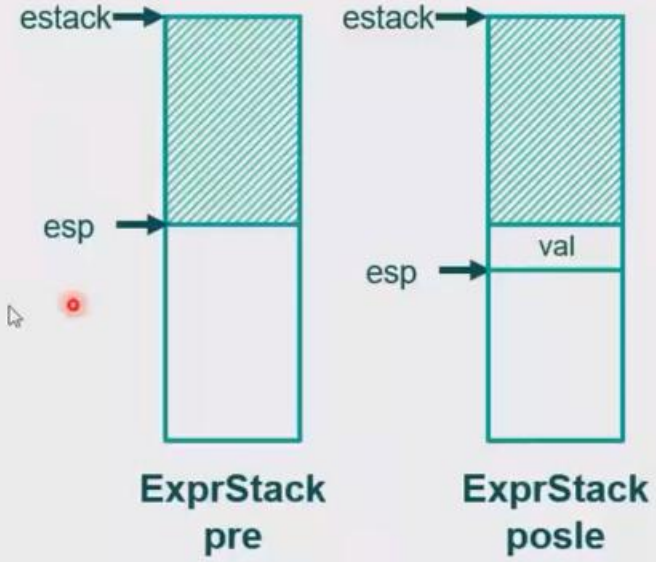
Mikrojava VM koristi sledeće instrukcije:

2. Instrukcije za load i store globalnih promenljivih

`getstatic s`

- Koristi se za stavljanje globalnih promenljivih na stek pri njihovom korišćenju u telu funkcija.
- **Primer:** učitavanje lokalne promenljive max na stek:

  `getstatic 0`




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

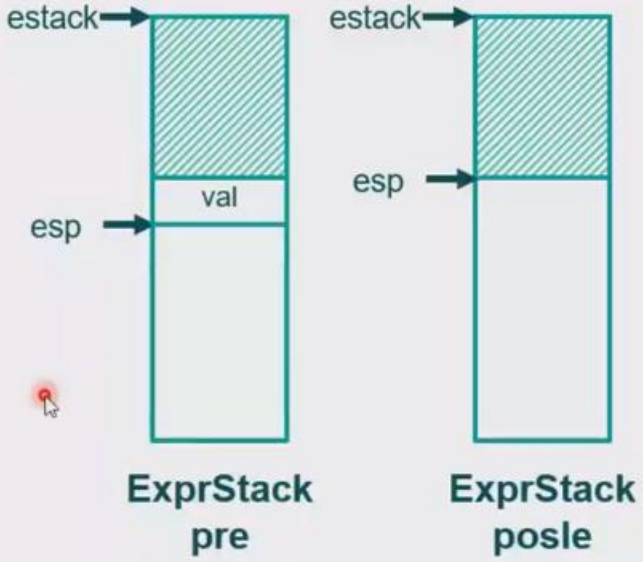
Mikrojava VM koristi sledeće instrukcije:

2. Instrukcije za load i store globalnih promenljivih

`putstatic s`

- Koristi se za smeštanje vrednosti sa steka u globalne promenljive prilikom njihovog korišćenja u telu funkcija.
- **Primer:** smeštanje vrednosti u promenljivu max:

  `putstatic 0`




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

3. Instrukcije za load i store polja objekata

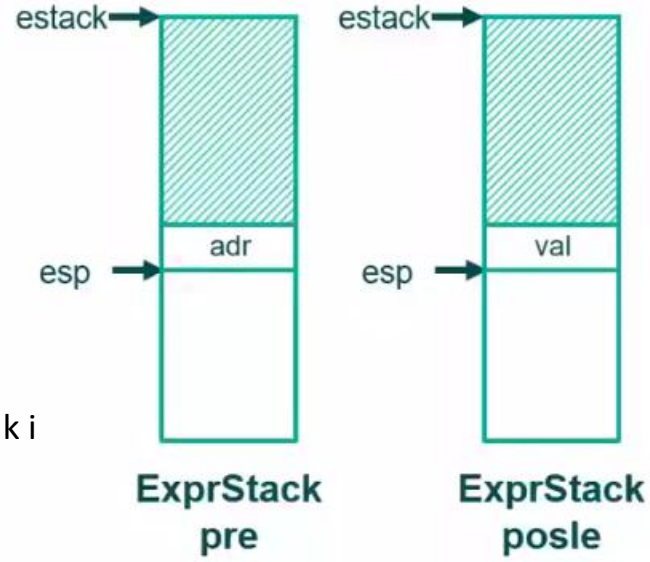
`getfield s`

- Koristi se za stavljanje polja objekata na stek pri njihovom korišćenju u telu funkcija.

- **Primer:** učitavanje polja f objekta k na stek:

  `load_2` Na stek stavljam k

  `getfield 1` Na stek skidam k i stavljam polje f objekta k, nije getfield 0 jer je nulta promenljiva this, a ne k




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

3. Instrukcije za load i store polja objekata

putfield s

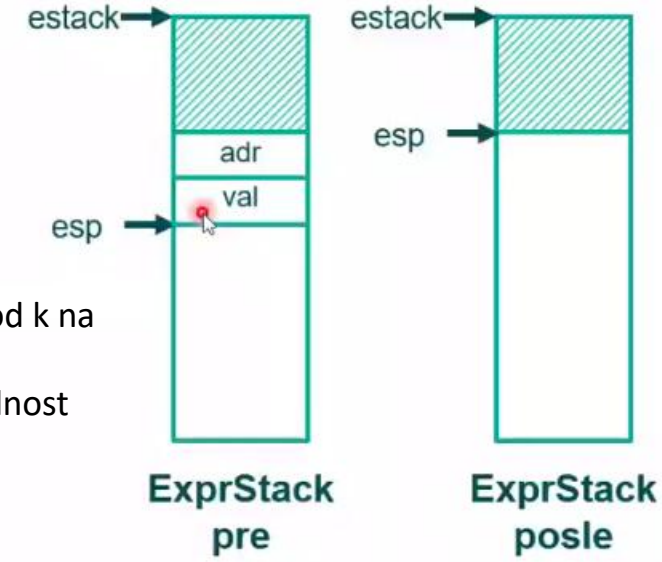
- Koristi se za smeštanje vrednosti sa steka u polje objekta prilikom njegovog korišćenja u telu funkcija.

- **Primer:** smeštanje vrednosti u polje f objekta k:

```
load_2
...
putfield 1
```

Smestam adresu od k na stek

Smesta se neka vrednost na stek




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

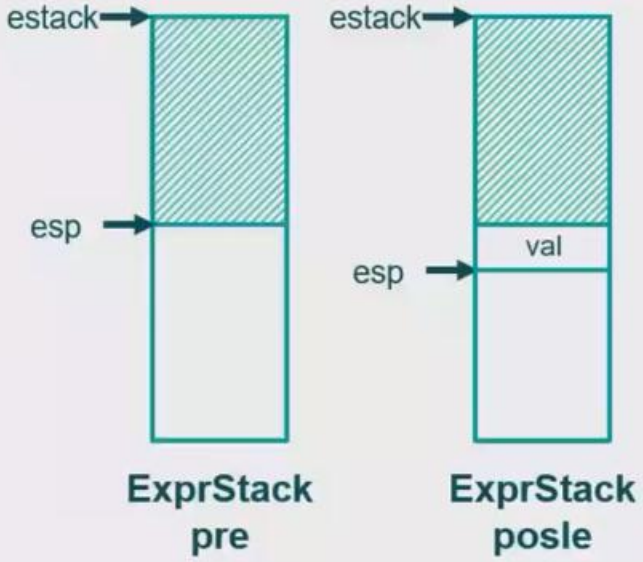
4. Instrukcije za load konstanti

const w

- Koristi se za stavljanje konstanti na stek pri njihovom korišćenju u telu funkcija. Za konstante 0-5 posebne instrukcije bez operanda: ***const_0***, ***const_1***, ***const_2***, ***const_3***, ***const_4***, ***const_5***. Za konstantu -1 posebna instrukcija ***const_m1***.

- **Primer:** učitavanje konstante N na stek:

```
const 2
(ili skraćeno) const_2
```




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

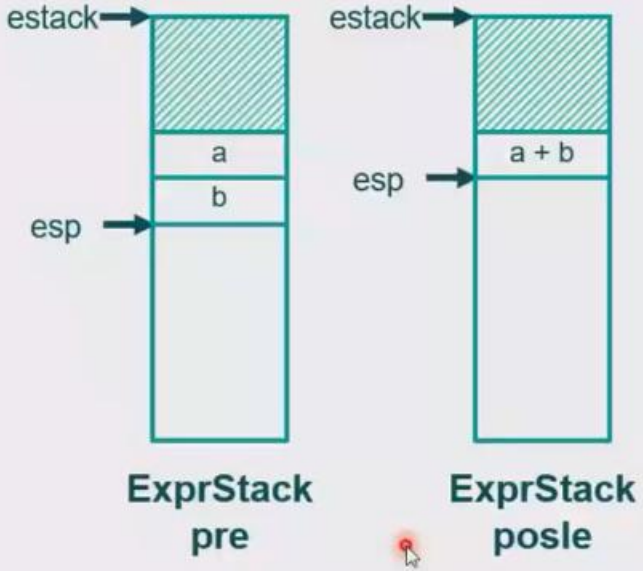
Mikrojava VM koristi sledeće instrukcije:

5. Aritmetičke intrukcije

add

- Koristi se za sabiranje dveju vrednosti koje su prethodno postavljene na stek. Ostale instrukcije: ***sub***, ***mul***, ***div*** i ***rem***.
- **Primer:** sabiranje lokalne promeljive x i konstante 1:

```
load_0
const_1
add
```




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

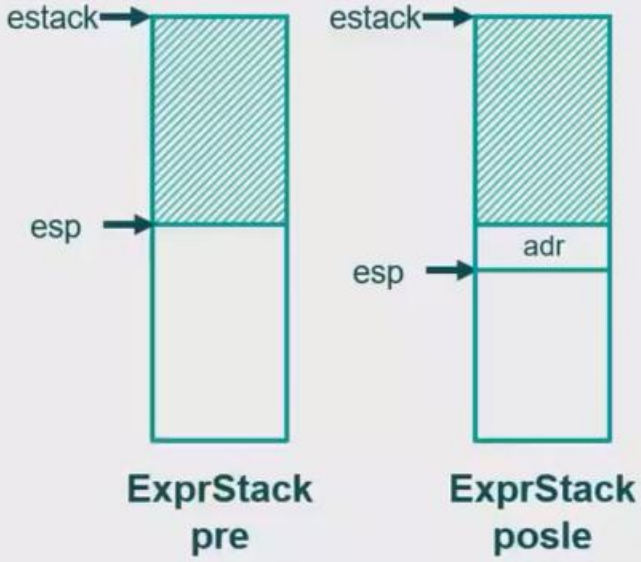
Mikrojava VM koristi sledeće instrukcije:

6. Instrukcije za dinamičku alokaciju memorije

new s

- Koristi se za alokaciju memorije na heap-u prilikom pravljena objekta. s – veličina objekta u broju bajtova.
- **Primer:** kreiranje novog objekta klase K:

new 8 2 reci(this I f), tj 8B




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

6. Instrukcije za dinamičku alokaciju memorije

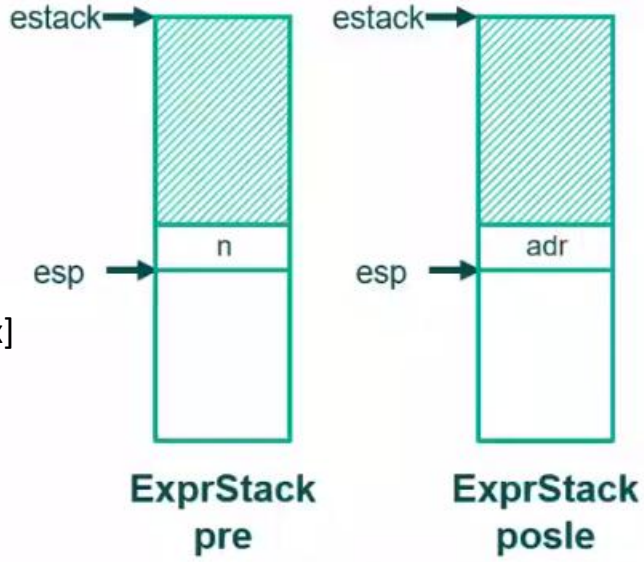
`newarray b`

- Koristi se za alokaciju memorije na heap-u prilikom pravljena nizova.
  b = (0 – niz bajtova, 1 – niz reči)

- **Primer:** kreiranje novog niza celih brojeva dužine N:

```
const_2
newarray 1
```

Da je bilo new int[x]
ovde bi islo load_0




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Žiža Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

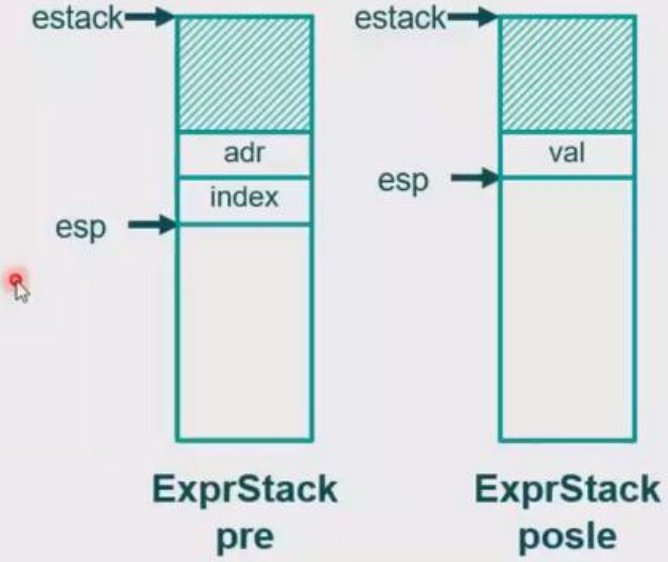
Mikrojava VM koristi sledeće instrukcije:

7. Instrukcije za load i store elemenata niza

aload

- Koristi se za postavljanje elemenata niza na stek prilikom njihovog korišćenja u telu funkcija. Posebna instrukcija za nizove bajtova: ***baload***
- **Primer:** dohvatanje prvog elementa niza niz:

```
load_1
const_1
aload
```




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

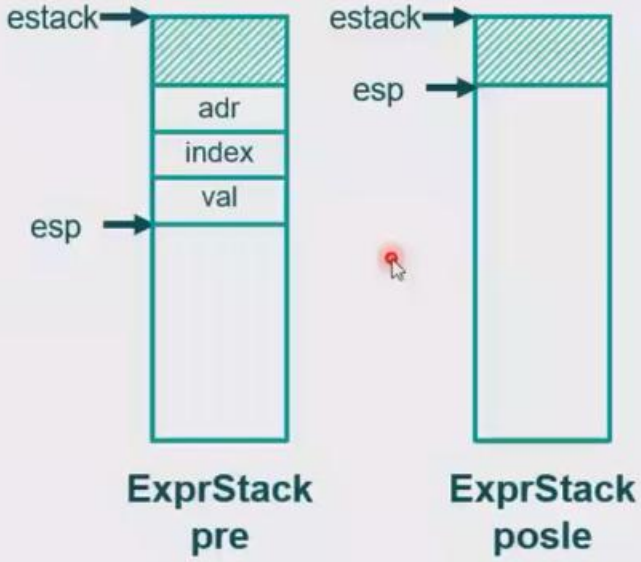
7. Instrukcije za load i store elemenata niza

astore

- Koristi se za smeštanje vrednosti sa steka u element niza prilikom njegovog korišćenja u telu funkcija. Za nizove bajtova: ***bastore***
- **Primer:** smeštanje vrednosti u nulti element niza niz:

```
load_1
const_0
...
astore
```

Stavljam neku vrednost na stek




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

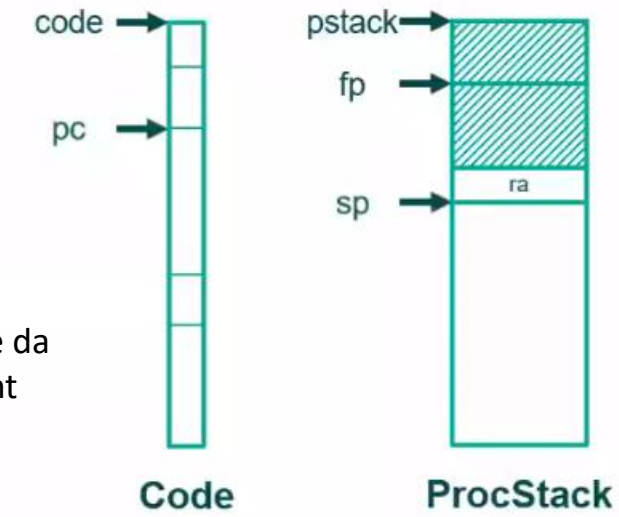
call s

- Koristi se za pozivanje funkcije koja se nalazi na pomeraju s od tekuće vrednosti PC-a.

- **Primer:** poziv metode put uz prosleđivanje argumenta:

```
load_2
getfield 1
call pom
```

Ova dva koraka je da bi stavili argument fje(f) na stek




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

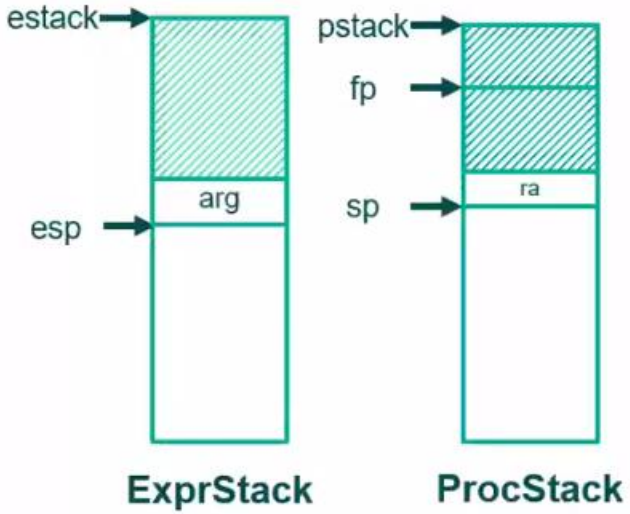
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

```
enter b1 b2
```

Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

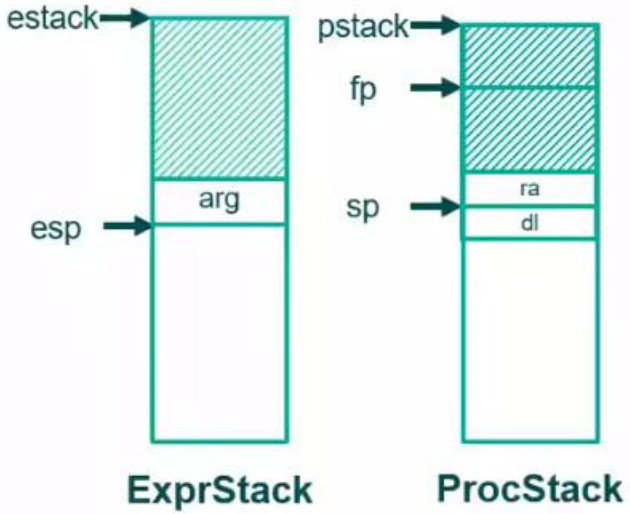
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

```
enter b1 b2
```

Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

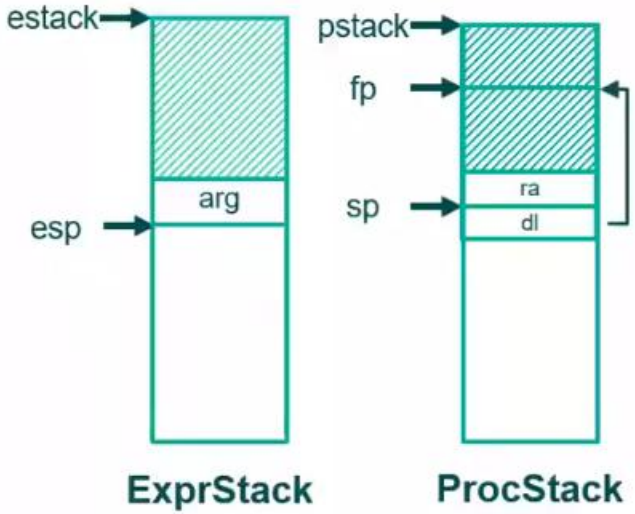
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

```
enter b1 b2
```

Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

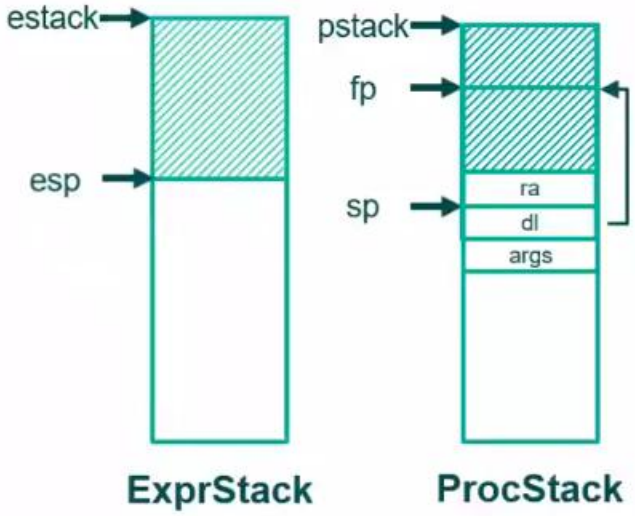
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

```
enter b1 b2
```

Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

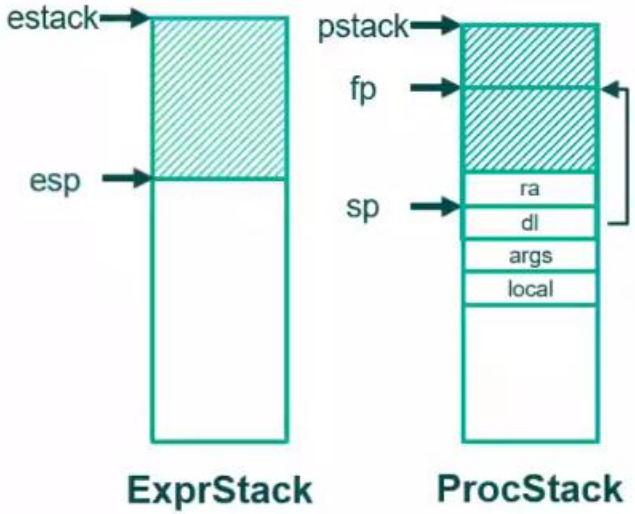
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

```
enter b1 b2
```

Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.



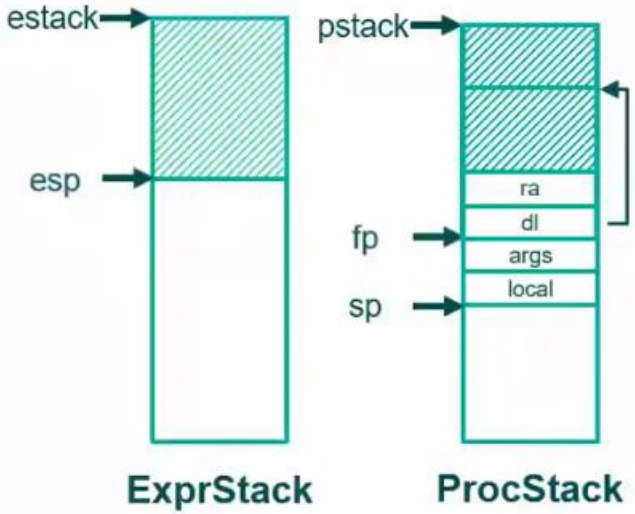
+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

```
enter b1 b2
```

Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

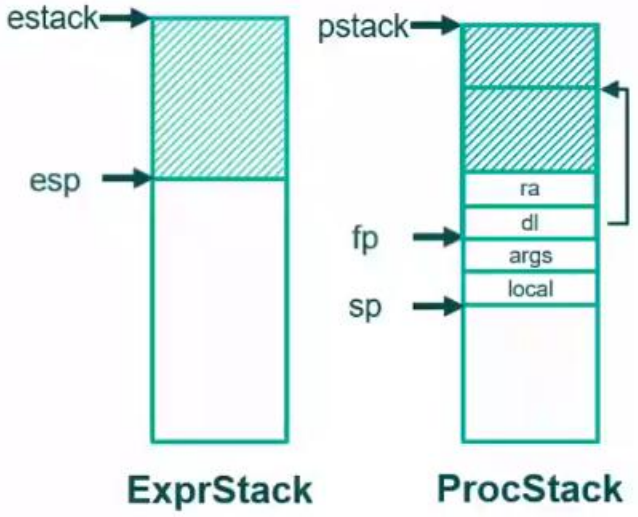
8. Instrukcije za rad sa funkcijama

`enter b1 b2`

- Pojavljuje se uvek kao prva instrukcija svake funkcije. b1 – broj argumenata, b2 – broj argumenata i lokalnih promenljivih. Inicijalizuje stek procedura.

- **Primer:** ulazak u metodu put (jedan argument – b1 = 1, jedan argument i jedna lokalna promenljiva – b2 = 2):

  `enter 1 2`




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

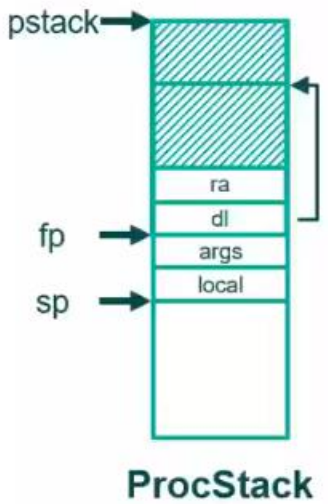
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`exit`

Pojavljuje se kao pretposlednja instrukcija svake funkcije. Oslobađa prostor za stvarne argumente i lokalne promenljive i restaurira polje kontrolne veze.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

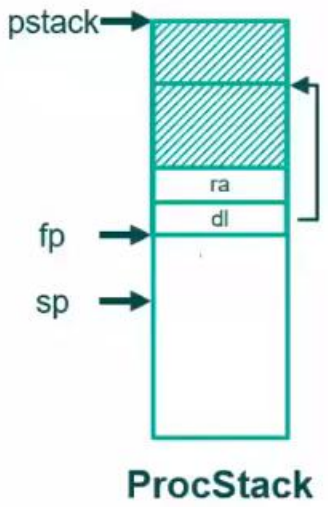
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`exit`

Pojavljuje se kao pretposlednja instrukcija svake funkcije. Oslobađa prostor za stvarne argumente i lokalne promenljive i restaurira polje kontrolne veze.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza
Павле Грковић
Урош Стоковић
Катарина Милошевић
Сретен Мандић
Радивоје Васиљевић
Софија Чоловић

Mikrojava VM

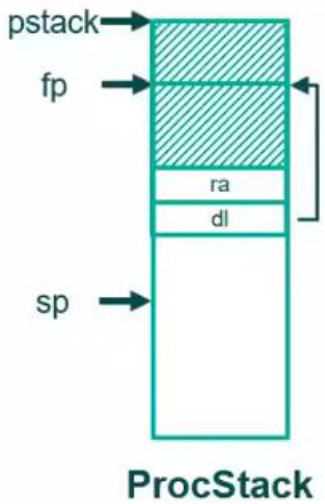
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

exit

Pojavljuje se kao pretposlednja instrukcija svake funkcije. Oslobađa prostor za stvarne argumente i lokalne promenljive i restaurira polje kontrolne veze.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

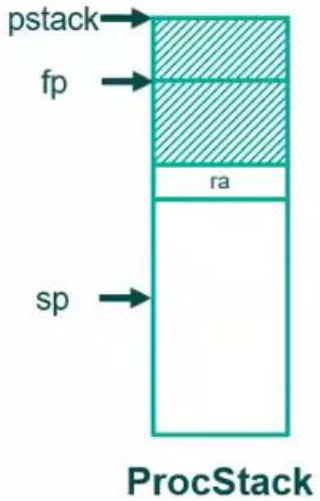
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

exit

Pojavljuje se kao pretposlednja instrukcija svake funkcije. Oslobađa prostor za stvarne argumente i lokalne promenljive i restaurira polje kontrolne veze.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza
Павле Грковић
Урош Стоковић
Катарина Милошевић
Сретен Мандић
Радивоје Васиљевић
Софија Чоловић

Mikrojava VM

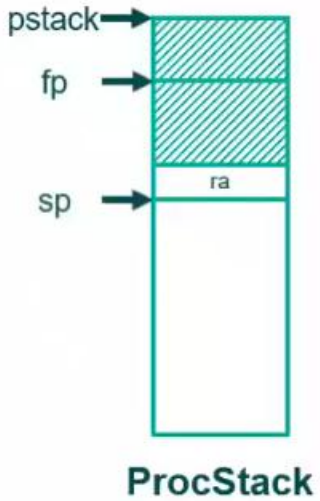
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`exit`

Pojavljuje se kao pretposlednja instrukcija svake funkcije. Oslobađa prostor za stvarne argumente i lokalne promenljive i restaurira polje kontrolne veze.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

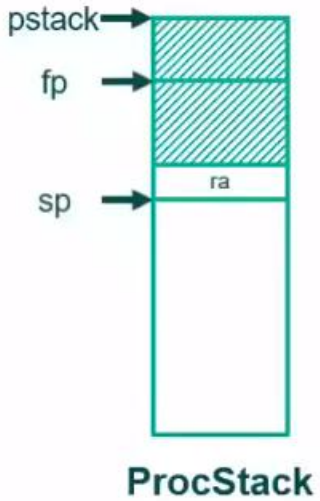
8. Instrukcije za rad sa funkcijama

exit

- Pojavljuje se kao pretposlednja instrukcija svake funkcije. Oslobađa prostor za stvarne argumente i lokalne promenljive i restaurira polje kontrolne veze.

- **Primer:** povratak iz funkcije put (uklanjaju se x i l sa steka procedura):

exit




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

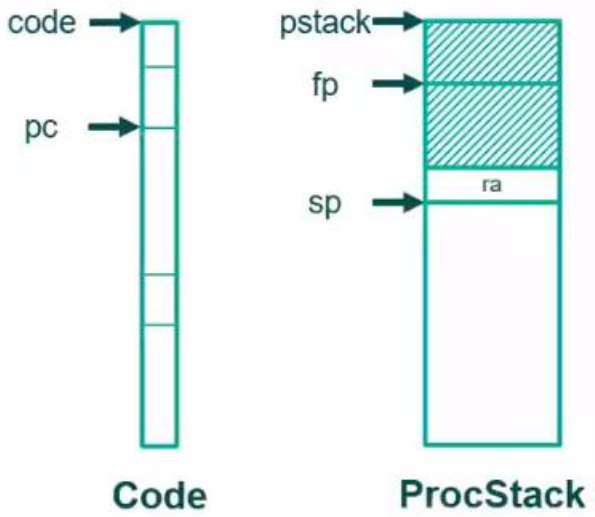
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`return`

Koristi se za povratak iz funkcije. Sa steka procedura skida povratnu adresu i dodeljuje je registru PC.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza
Павле Грковић
Урош Стоковић
Катарина Милошевић
Сретен Мандић
Радивоје Васиљевић
Софија Чоловић

Mikrojava VM

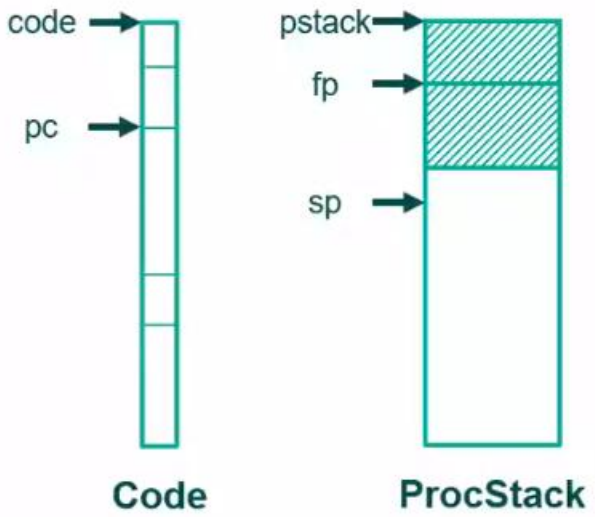
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`return`

Koristi se za povratak iz funkcije. Sa steka procedura skida povratnu adresu i dodeljuje je registru PC.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

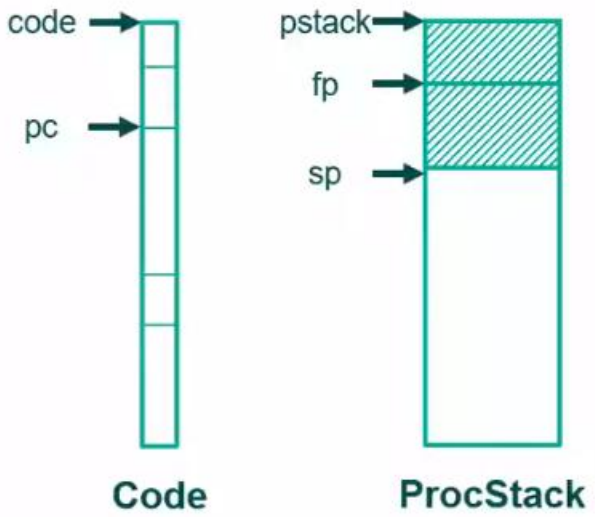
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`return`

Koristi se za povratak iz funkcije. Sa steka procedura skida povratnu adresu i dodeljuje je registru PC.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

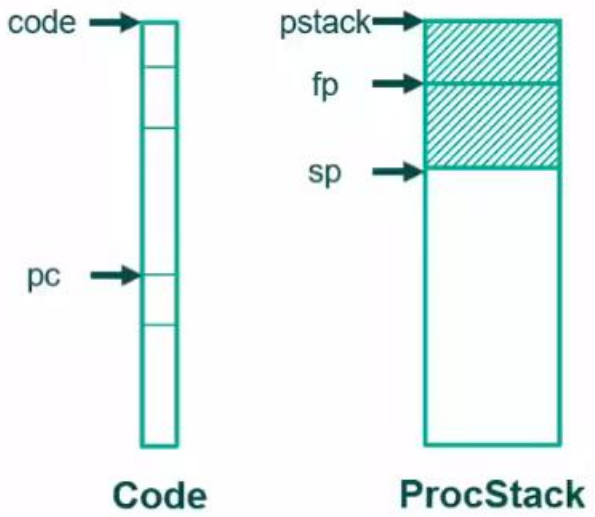
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`return`

Koristi se za povratak iz funkcije. Sa steka procedura skida povratnu adresu i dodeljuje je registru PC.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

instrukcije MJVM

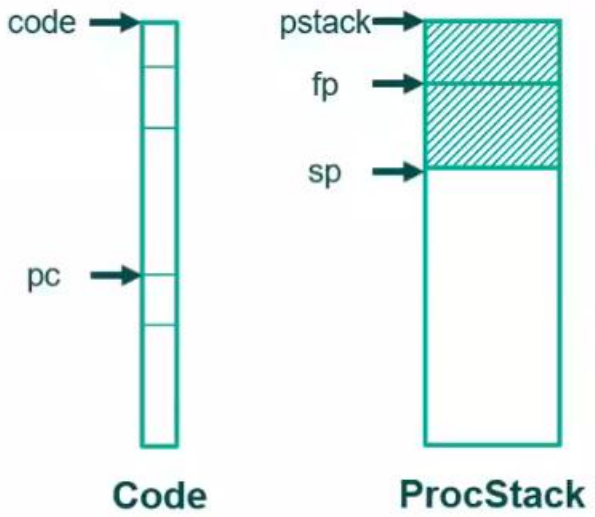
Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`return`

- Koristi se za povratak iz funkcije. Sa steka procedura skida povratnu adresu i dodeljuje je registru PC.
- **Primer:** povratak iz metode put (povratna vrednost ostaje na steku izraza):

```
load_0
exit
return
```




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

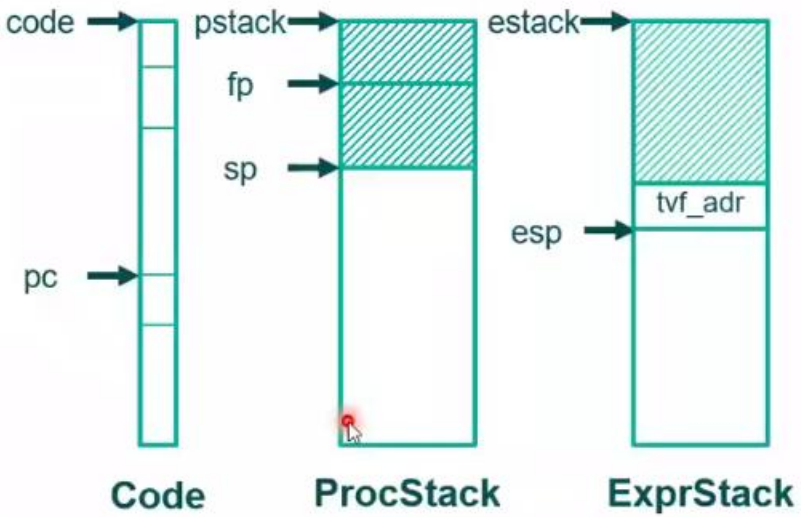
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`invokevirtual` w_1 $w_2 \ldots w_{n+1}$

Koristi se za polimorfno pozivanje metoda. Ime metode je specificirano uz instrukciju, a završava se sa -1.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

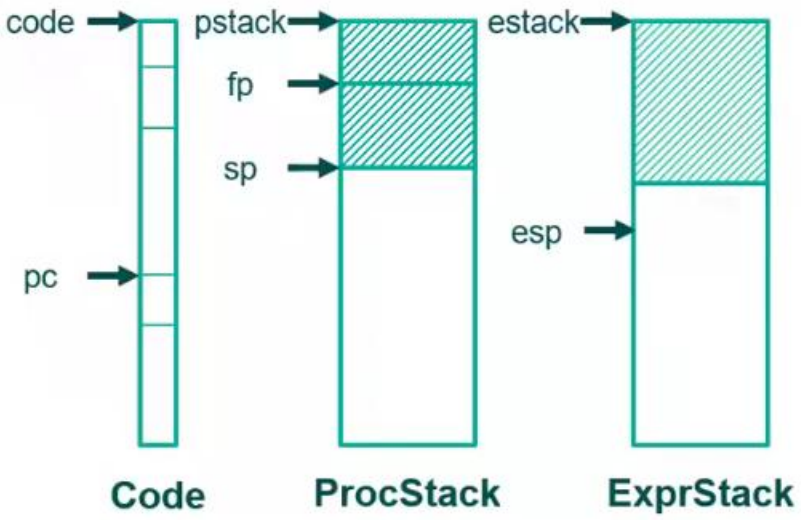
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`invokevirtual` $w_1\ w_2 \ldots w_{n+1}$

Koristi se za polimorfno pozivanje metoda. Ime metode je specificirano uz instrukciju, a završava se sa -1.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

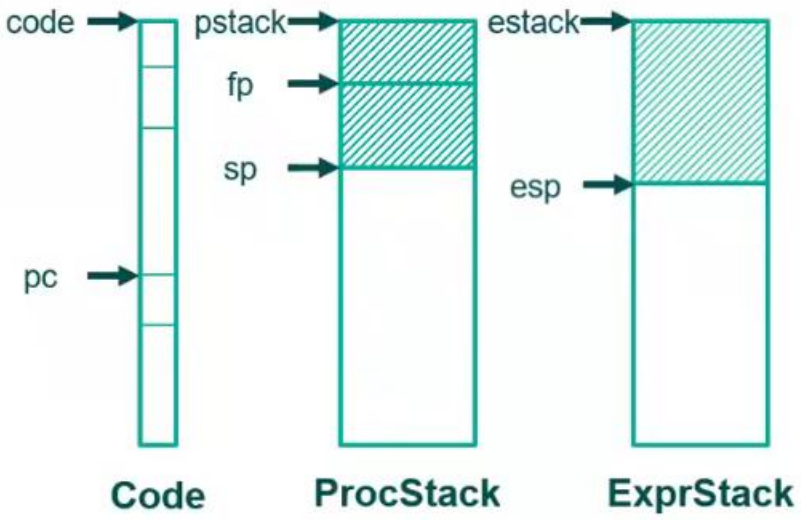
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`invokevirtual` $w_1\ w_2 \ldots w_{n+1}$

Koristi se za polimorfno pozivanje metoda. Ime metode je specificirano uz instrukciju, a završava se sa -1.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

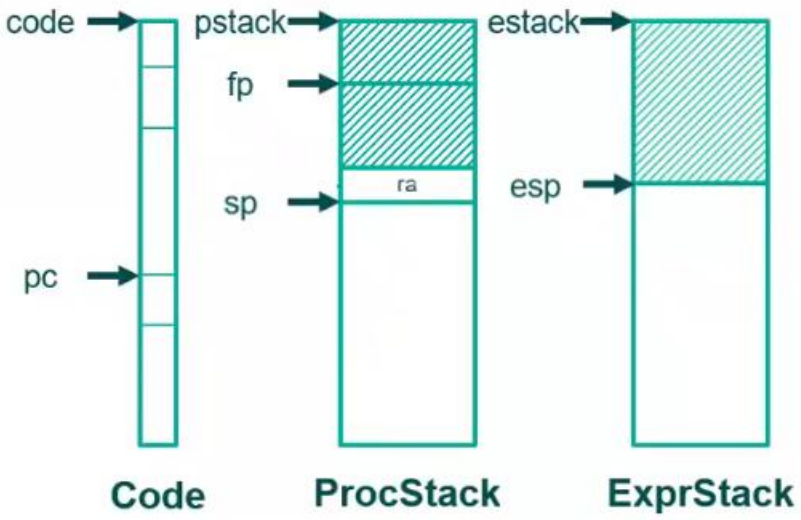
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`invokevirtual` $w_1\ w_2 \ldots w_{n+1}$

Koristi se za polimorfno pozivanje metoda. Ime metode je specificirano uz instrukciju, a završava se sa -1.



+10 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

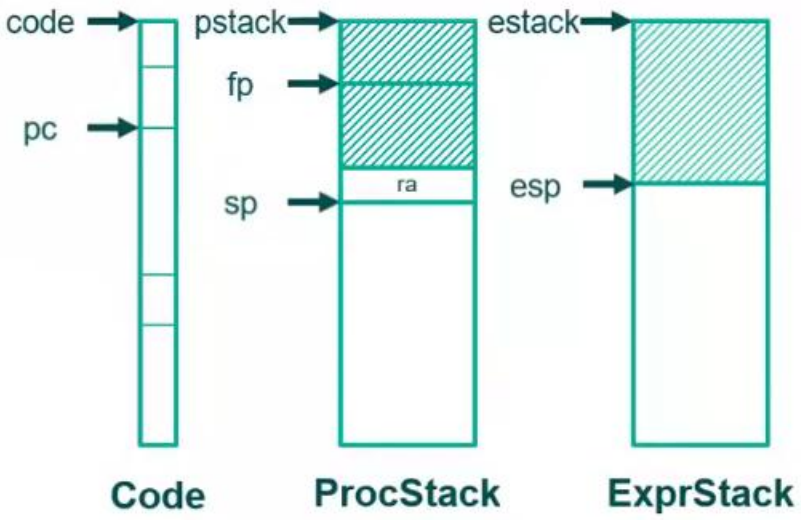
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

8. Instrukcije za rad sa funkcijama

`invokevirtual` $w_1\ w_2 \ldots w_{n+1}$

Koristi se za polimorfno pozivanje metoda. Ime metode je specificirano uz instrukciju, a završava se sa -1.



+10 МН НМ ФС KZ УС КМ СМ РВ СЧ
Kristijan Ziza Павле Грковић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Mikrojava VM

instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

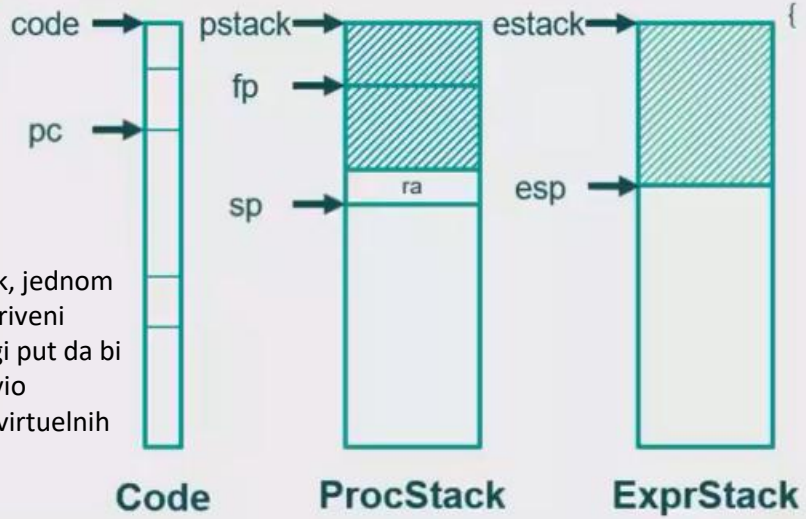
8. Instrukcije za rad sa funkcijama

invokevirtual w_1 $w_2 \ldots w_{n+1}$

- Koristi se za polimorfno pozivanje metoda. Ime metode je specificirano uz instrukciju, a završava se sa -1.
- **Primer:** polimorfni poziv metode m objekta k:

```
load_2
load_2
getfield 0
invokevirtual 'm' -1
```

Dva puta load-ujem k, jednom jer metoda m ima skriveni argument this,a drugi put da bi pomocu getfield stavio pokazivac na tabelu virtuelnih funkcija




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+9 МН НМ ФС KZ Kristijan Ziza Павле Грковић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

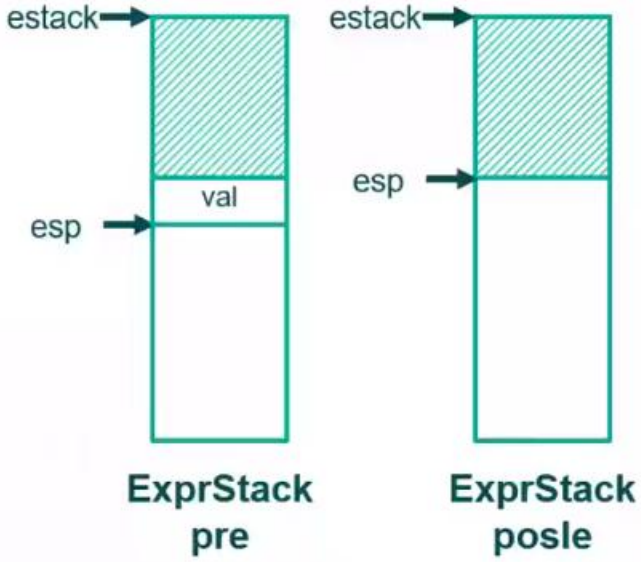
Mikrojava VM koristi sledeće instrukcije:

9. Instrukcije za rad sa stekom

pop

- Koristi se za skidanje elementa sa vrha steka izraza. Skinuta vrednost nije dostupna.
- **Primer:** skidanje sa steka neiskorišćene povratne vrednosti funkcije put:

```
...
call pom
pop
```




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

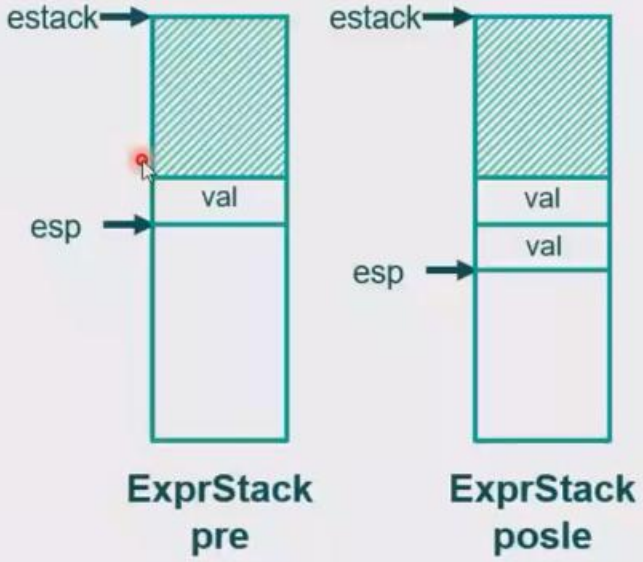
Mikrojava VM koristi sledeće instrukcije:

9. Instrukcije za rad sa stekom

dup

- Koristi se za dupliranje elementa koji se nalazi na vrhu steka izraza. Postoje još i varijante ***dup2***, ***dup_x1*** i ***dup_x2***.
- **Primer:** dupliranje k prilikom polimorfnog poziva (jedno za argument this, drugo za dohvatanje adrese *TVF*:

```
load_2
dup
...
```




```
program ABC
int max;
const int N = 2;
class K{
      int f;
      { void m(){} }
}
{
      int put (int x) int l; {
            return x;
      }
      void main()
            int x; int niz[]; K k;
      {
            niz = new int[N];
            k = new K;
            k.f = x + 1;
            max = 1;
            niz[0] = niz[1] + max;
            put(k.f);
            k.m();
      }
}
```


+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

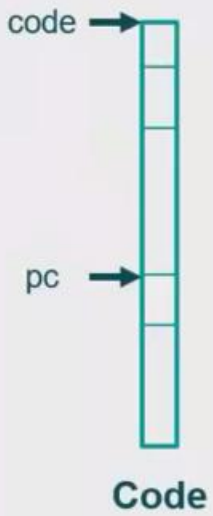
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

`jmp s`

Koristi se za bezuslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a.



+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

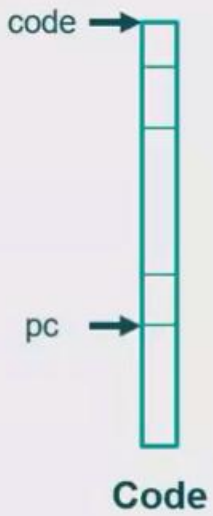
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

`jmp s`

Koristi se za bezuslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a.



+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

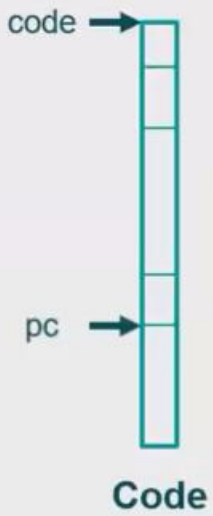
Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

`jmp s`

- Koristi se za bezuslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a.
- **Primer:** preskakanje else grane:

```
const_1
store_0
jmp iza_else
```




```
program ABC
const int N = 2;
{
    void main()
        int x;
    {
        if(N != 0){
            x = 1;
            // ***
        }else{
            x = 2;
        }
    }
}
```


+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

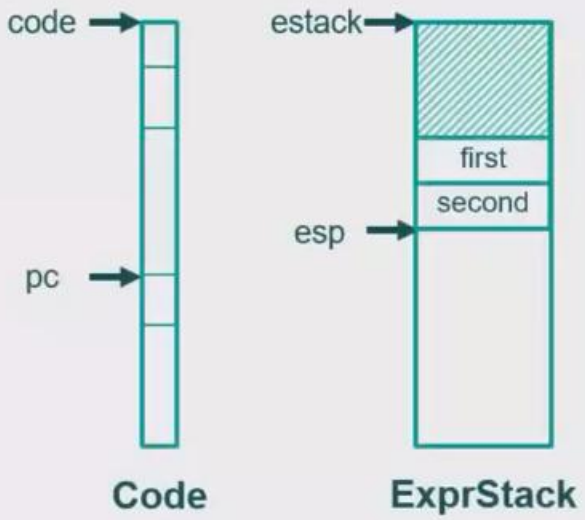
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

`jeq s`

Koristi se za uslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a. Skok se vrši ukoliko su dva elementa koja se nalaze na vrhu steka izraza jednaka. Postoje i: ***jne***, ***jlt***, ***jle***, ***jgt*** i ***jge***.



+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

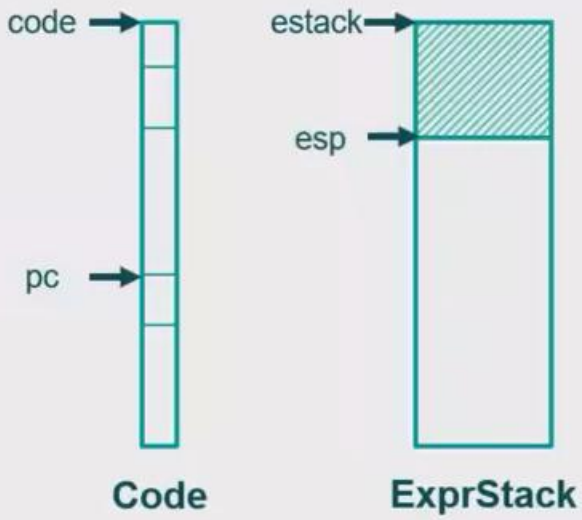
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

`jeq s`

Koristi se za uslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a. Skok se vrši ukoliko su dva elementa koja se nalaze na vrhu steka izraza jednaka. Postoje i: ***jne***, ***jlt***, ***jle***, ***jgt*** i ***jge***.



+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

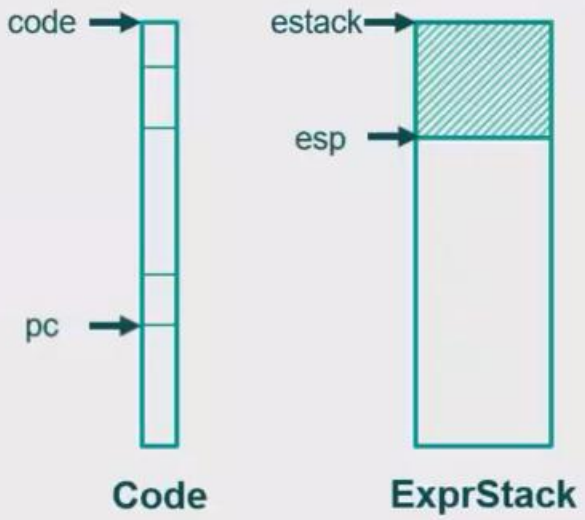
instrukcije MJVM

Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

```
jeq s
```

Koristi se za uslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a. Skok se vrši ukoliko su dva elementa koja se nalaze na vrhu steka izraza jednaka. Postoje i: ***jne***, ***jlt***, ***jle***, ***jgt*** i ***jge***.



+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

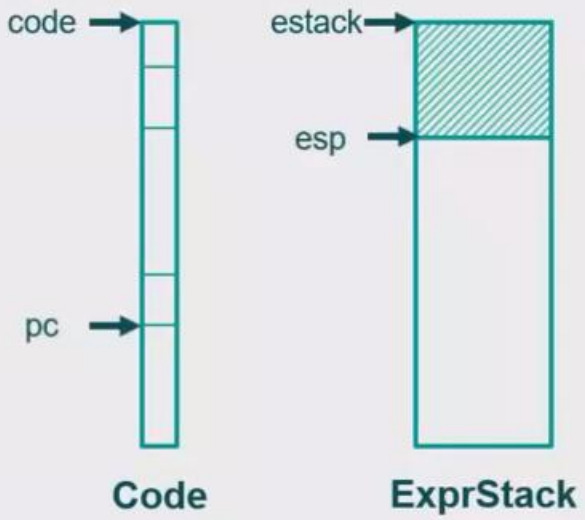
Mikrojava VM koristi sledeće instrukcije:

10. Instrukcije za rad sa skokovima

`jeq s`

- Koristi se za uslovni skok. *s* je pomeraj koji se dodaje na trenutnu vrednost **PC**-a. Skok se vrši ukoliko su dva elementa koja se nalaze na vrhu steka izraza jednaka. Postoje i: ***jne***, ***jlt***, ***jle***, ***jgt*** i ***jge***.
- **Primer:** preskakanje then grane:

```
const_2
const_0
jeq na_else
```




```
program ABC
const int N = 2;
{
    void main()
        int x;
    {
        if(N != 0){
            x = 1;
            // ***
        }else{
            x = 2;
        }
    }
}
```


+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

instrukcije MJVM

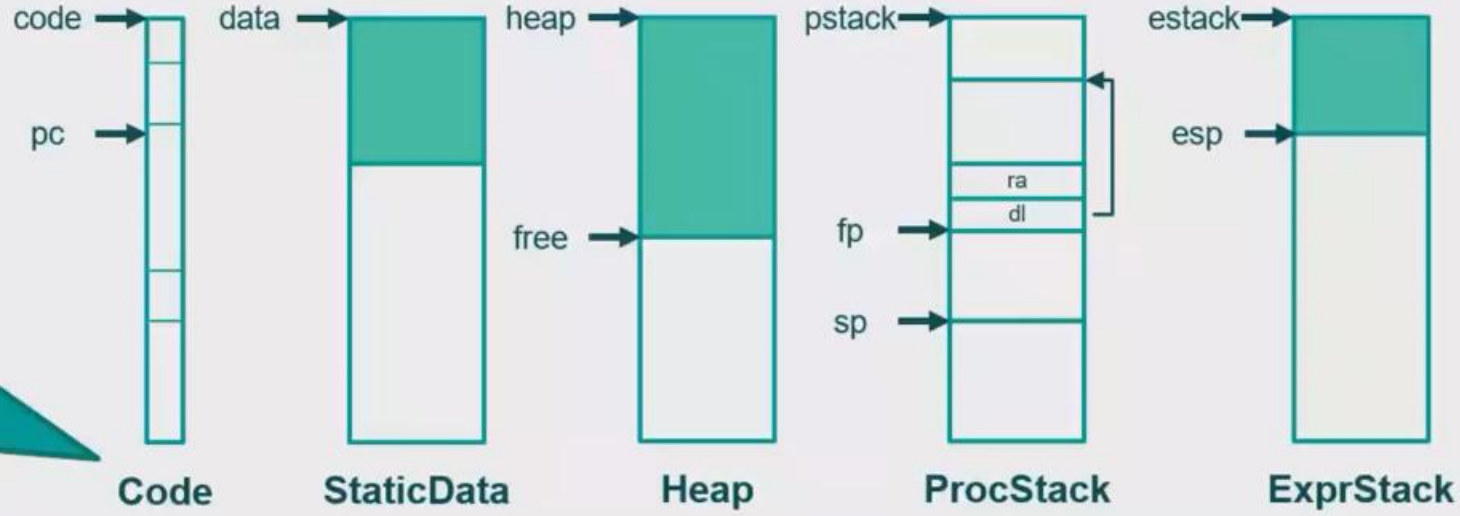
Mikrojava VM koristi još i sledeće instrukcije:

- Instrukcije za ulaz/izlaz: ***read, print, bread, bprint***
- Instrukcija za negaciju: ***neg***
- Pomeračke instrukcije: ***shl, shr***
- Instrukcija za inkrementiranje: ***inc***
- Instrukcija za dohvatanje dužine niza: ***arraylength***
- Instrukcija za generisanje runtime greške: ***trap***

+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Mikrojava VM

```
load, store, getstatic,
putstatic, getfield,
putfield, const, add,
sub, mul, div, rem,
neg, shl, shr, inc,
new, newarray, aload,
astore, baload,
bastore, arraylength,
pop, dup, dup2, jmp,
jeq, jne, jle, jlt,
jge, jgt, call, return,
enter, exit, read,
print, bread, bprint,
trap, invokevirtual,
dup_x1, dup_x2
```



+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

Napisati MJ bajtkod koji odgovara telu metode *main* bez *enter*, *exit* i *return* sekvenci.

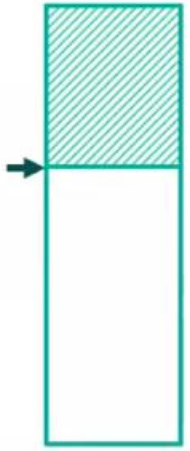
Za svaku liniju izvornog koda naznačiti koji deo bajtkoda joj pripada.

```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        y++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


+9 МН НМ KZ ФС KZ Kristijan Ziza Стефан Теслић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```

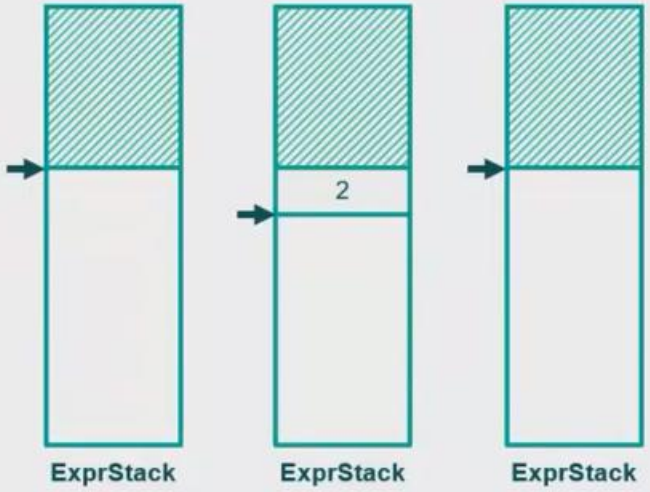


ExprStack

+8 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
const_2
putstatic 0
```

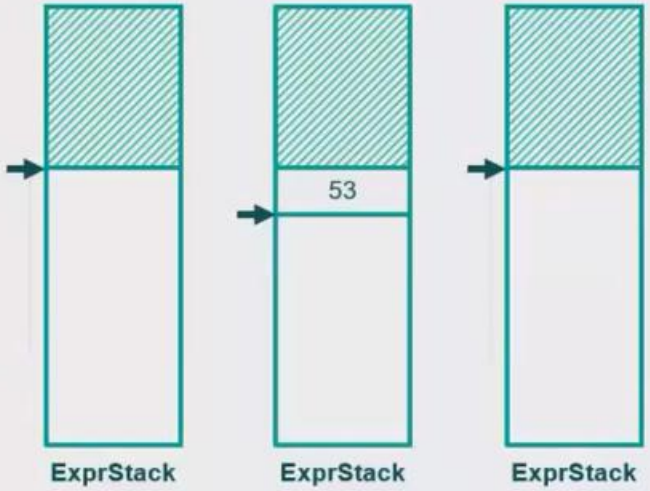



```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


+8 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
const 53
store_1
```

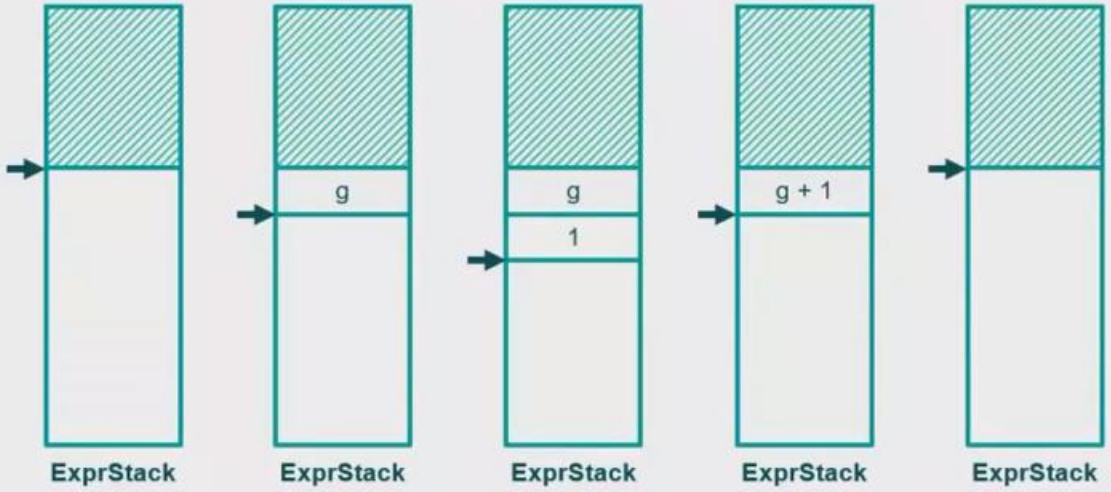



```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


+8 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
getstatic 0
const_1
add
putstatic 0
```



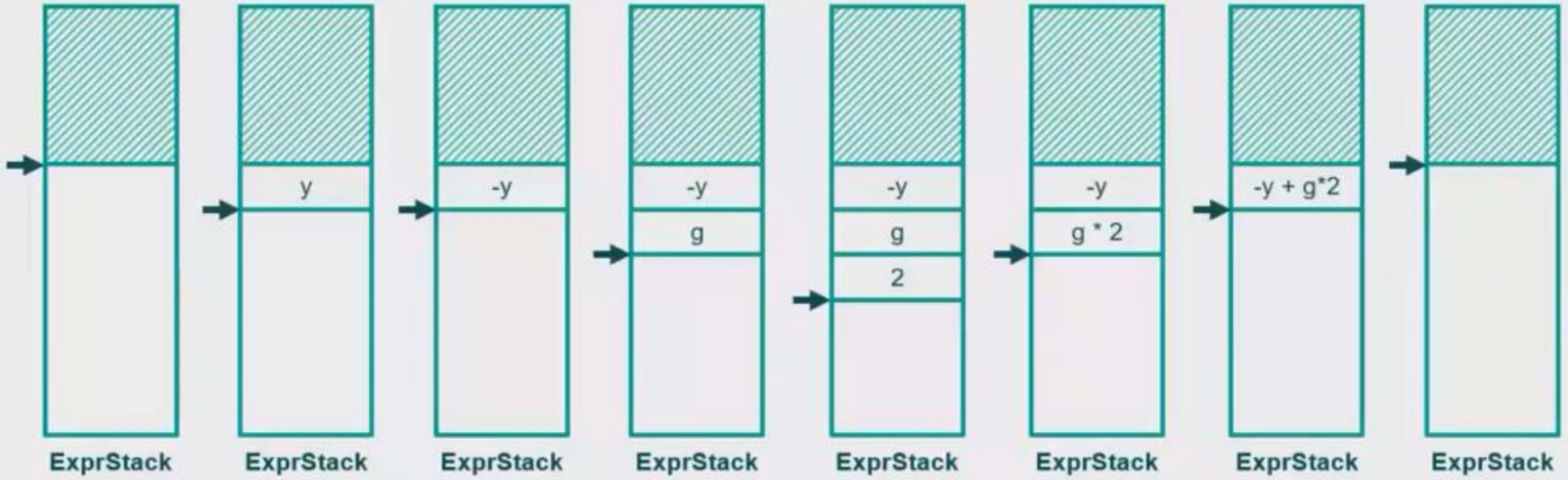

```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


+8 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
load_1
neg
getstatic 0
const_2
mul
add
store_0
```

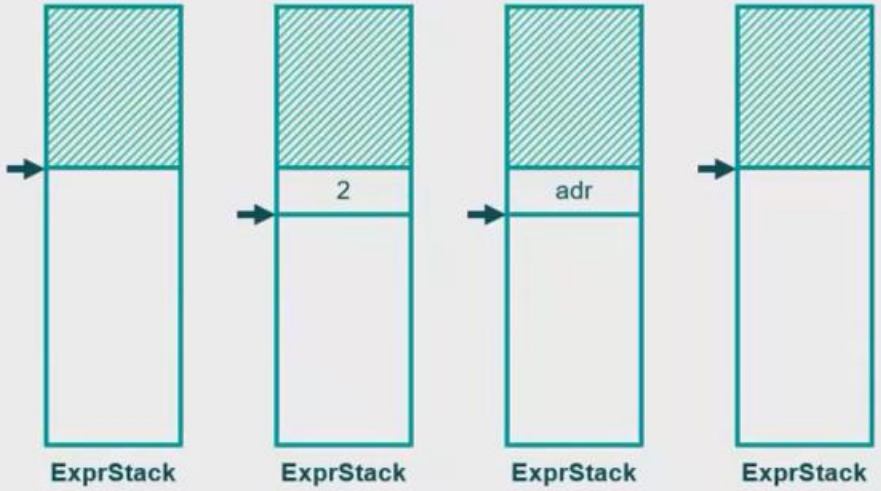
```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```



+8 МН НМ ФС KZ КБ УС КМ СМ РВ СЧ
Kristijan Ziza
Ксенија Булатовић
Урош Стоковић
Катарина Милошевић
Сретен Мандић
Радивоје Васиљевић
Софија Чоловић

Zadatak 1.

```
const_2
newarray 1
store_2
```

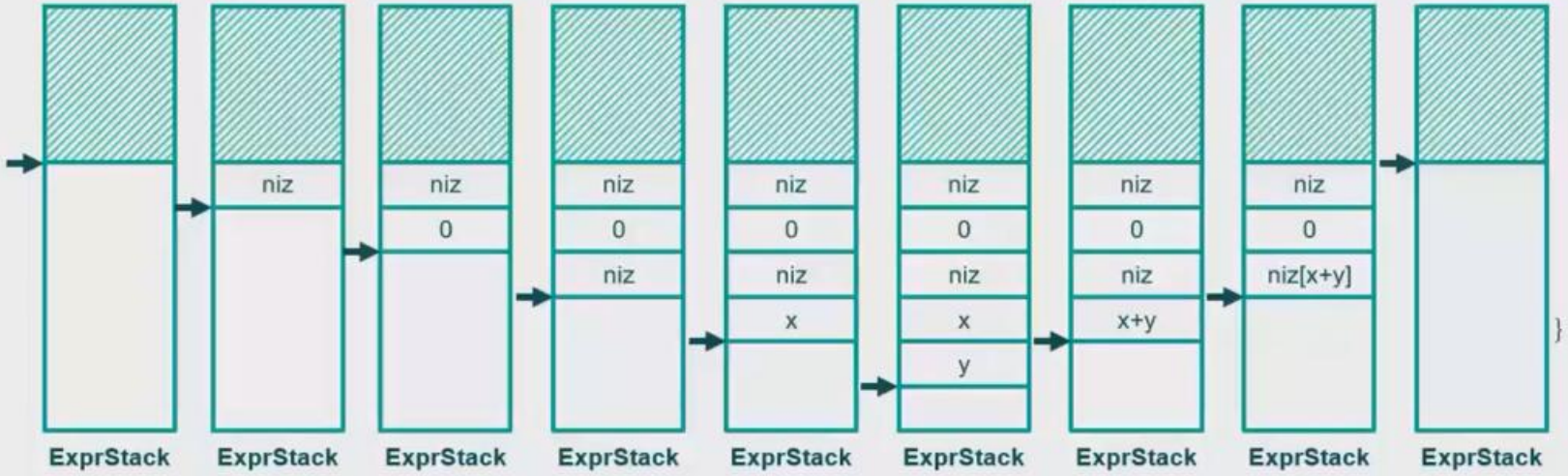



```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


+8 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
load_2
const_0
load_2
load_0
load_1
add
aload
astore
```

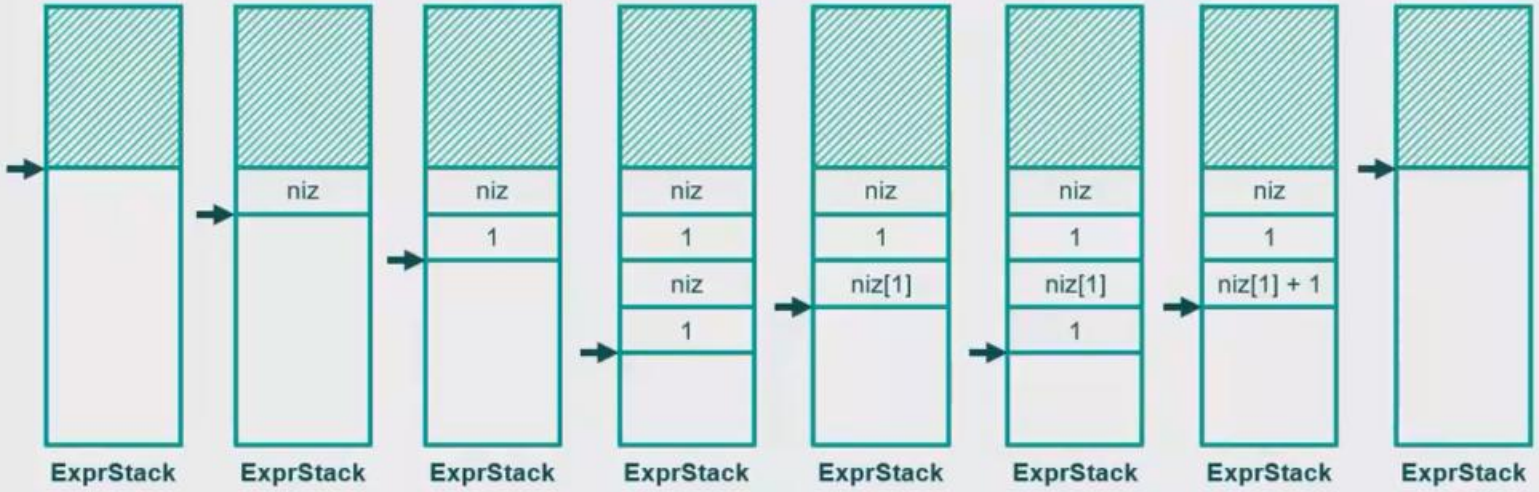



```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


1:16:38 / 2:02:16

Zadatak 1.

```
load_2
const_1
dup2
aload
const_1
add
astore
```




```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 1.

```
const_2
putstatic 0
```

```
const 53
store_1
```

```
getstatic 0
const_1
add
putstatic 0
```

```
load_1
neg
getstatic 0
const_2
mul
add
store_0
```

```
program ABC
const int N = 2;
int g;
{
    void main()
        int x, y;
        int niz[];
    {
        g = N;
        y = 53;
        g++;
        x = -y + g * N;
        niz = new int[N];
        niz[0] = niz[x+y];
        niz[1]++;
    }
}
```

```
const_2
newarray 1
store_2
```

```
load_2
const_0
load_2
load_0
load_1
add
aload
astore
```

```
load_2
const_1
dup2
aload
const_1
add
astore
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 2.

Napisati MJ bajtkod koji odgovara telu metode *main* bez *enter*, *exit* i *return* sekvenci.

Za svaku liniju izvornog koda naznačiti koji deo bajtkoda joj pripada.

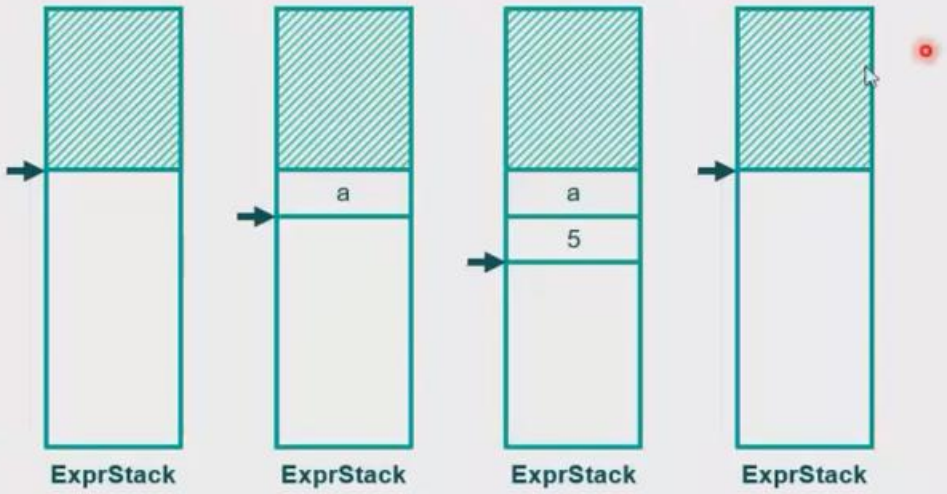
```
program ABC
class A{
    int i;
}
class B{
    A aniz[];
}
class C{
    B b;
}
{
    void main()
        C cniz[]; C c; A a;
    {
        a.i = 5;
        a.i++;
        cniz[0].b.aniz[1].i = 2;
        a = c.b.aniz[4];
    }
}
```


+9 МН НМ ФС KZ КБ УС КМ СМ РВ СЧ
Kristijan Ziza Ксенија Булатовић Урош Стоковић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Zadatak 2.

```
load_2
const_5
putfield 1
```

0 je rezervisano za pokazivac na tabelu virtuelnih funkcija, pa je int I u stvari polje pod brojem 1

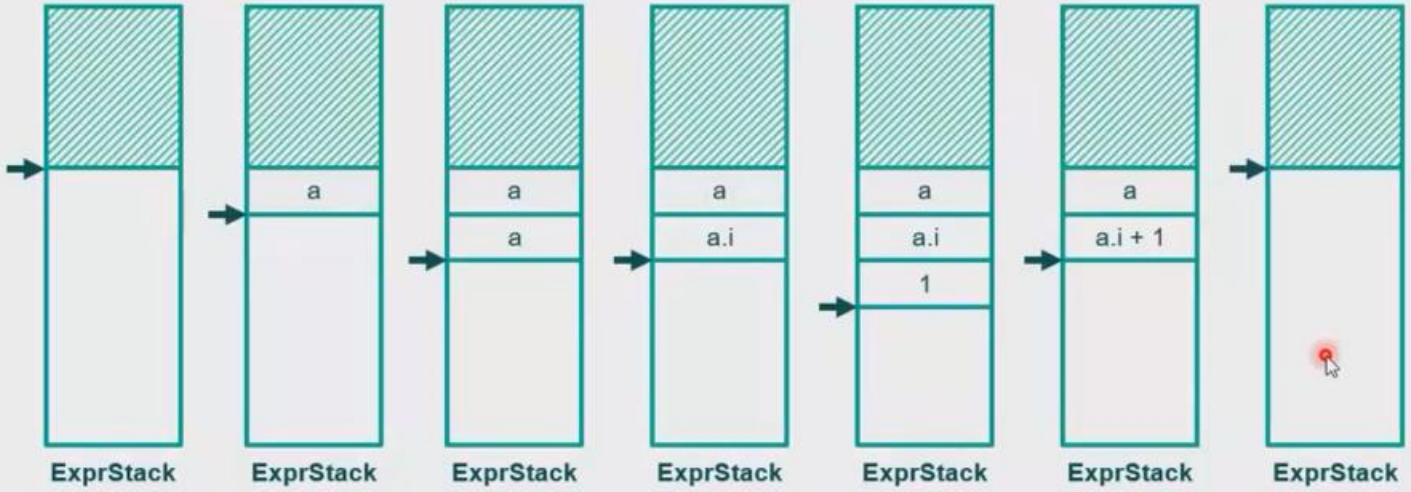



```
program ABC
class A{
    int i;
}
class B{
    A an[];
}
class C{
    B b;
}
{
    void main()
        C cn[]; C c; A a;
    {
        a.i = 5;
        a.i++;
        c.b.an[1].i = 2;
        a = cn[3].b.an[4];
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 2.

```
load_2
load_2
getfield 1
const_1
add
putfield 1
```

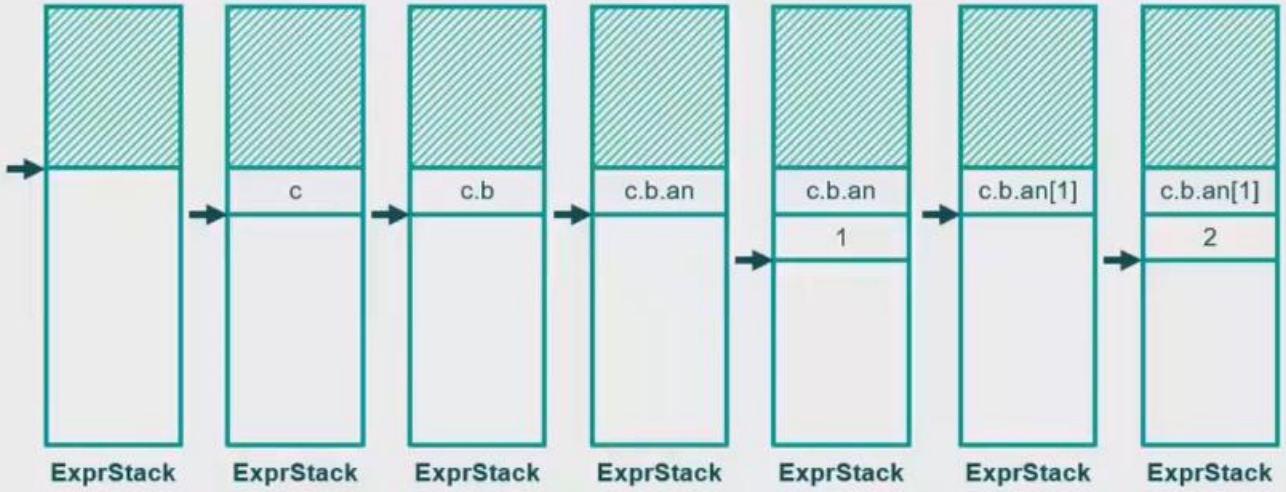



```
program ABC
class A{
    int i;
}
class B{
    A an[];
}
class C{
    B b;
}
{
    void main()
        C cn[]; C c; A a;
    {
        a.i = 5;
        a.i++;
        c.b.an[1].i = 2;
        a = cn[3].b.an[4];
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 2.

```
load_1
getfield 1
getfield 1
const_1
aload
const_2
putfield 1
```

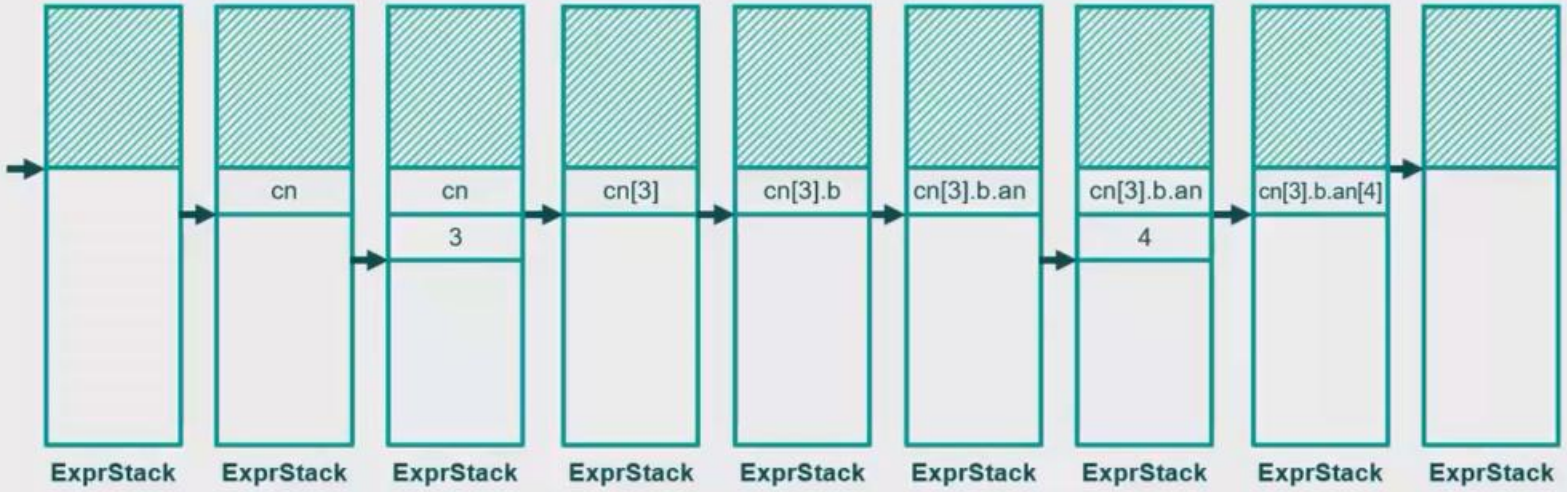



```
program ABC
class A{
    int i;
}
class B{
    A an[];
}
class C{
    B b;
}
{
    void main()
        C cn[]; C c; A a;
    {
        a.i = 5;
        a.i++;
        c.b.an[1].i = 2;
        a = cn[3].b.an[4];
    }
}
```


+9 МН НМ ФС KZ КБ УС КМ СМ РВ СЧ
Kristijan Ziza
Ксенија Булатовић
Урош Стоковић
Катарина Милошевић
Сретен Мандић
Радивоје Васиљевић
Софија Чоловић

Zadatak 2.

```
load_0
const_3
aload
getfield 1
getfield 1
const_4
aload
store_2
```




```
program ABC
class A{
    int i;
}
class B{
    A an[];
}
class C{
    B b;
}
{
    void main()
        C cn[]; C c; A a;
    {
        a.i = 5;
        a.i++;
        c.b.an[1].i = 2;
        a = cn[3].b.an[4];
    }
}
```


1:31:11 / 2:02:16

Zadatak 3.

Napisati MJ bajtkod koji odgovara telu metode *main* bez *enter*, *exit* i *return* sekvenci.

Za svaku liniju izvornog koda naznačiti koji deo bajtkoda joj pripada.

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
if       →   0: load_0
             1: const_1
             2: jne ???      ← na else granu
then     →   5: const_2
             6: store_1
             7: jmp ???      ← iza else grane
else     →  10: const_3
            11: store_1
iza else →  12: ...
```

jne instrukcija je 3B, 1 za kod instrukcije(jne), 2 za pomeraj

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
if      →   0: load_0
            1: const_1
            2: jne 8        ← na else granu
then    →   5: const_2
            6: store_1
            7: jmp 5        ← iza else grane
else    →  10: const_3
           11: store_1
iza else →  12: ...
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
for          12: const_0
             13: store_0
uslov        14: load_0
             15: const_3
             16: jge ???      izlazak iz for petlje
telo petlje  19: load_0
             20: const_4
             21: print
akcija nakon 22: inc 0 1
iteracije    25: jmp -11
```

Ovo samo govori na kojoj sirini treba print da ispise(bzv je stavio 4)

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

for →
uslov →
telo petlje →
akcija nakon iteracije →

```
12: const_0
13: store_0
14: load_0
15: const_3
16: jge 12
19: load_0
20: const_4
21: print
22: inc 0 1
25: jmp -11
28: ...
```

← izlazak iz for petlje

← povratak na proveru uslova

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+9 МН НМ ФС KZ Kristijan Ziza КБ Ксенија Булатовић УС Урош Стоковић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
while  →  28: load_0
          29: const_4
          30: jge ???      ← iza petlje
   if  →  33: load_0
          34: const_1
          35: jne ???      ← na else if
          38: jmp -10      ← na while
else if →  41: load_0
          42: const_2
          43: jne ???      ← na else
          46: jmp ???      ← iza petlje
  else →  49: const_0
          50: store_1
          51: load_1
          52: const_2
          53: rem
          54: store_1
          55: jmp -27      ← na while
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+7 МН НМ УС KZ Kristijan Ziza КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
while      28: load_0
           29: const_4
           30: jge 28        iza petlje
if         33: load_0
           34: const_1
           35: jne 6         na else if
           38: jmp -10       na while
else if    41: load_0
           42: const_2
           43: jne 6         na else
           46: jmp 12        iza petlje
else       49: const_0
           50: store_1
           51: load_1
           52: const_2
           53: rem
           54: store_1
           55: jmp -27       na while
iza while  58: ...
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+7 МН НМ УС KZ КБ ВТ КМ СМ РВ СЧ
Kristijan Ziza Ксенија Булатовић Војислав Томашевић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Zadatak 3.

```
if              58: load_0
                59: const_0
                60: jle ???      iza then grane
drugi uslov     63: load_1
                64: const_2
                65: jge ???      iza then grane
then grana      68: const_2
                69: store_0
iza then        70: ...
grane
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+8 МН НМ УС KZ Kristijan Žiža КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
if            58: load_0
              59: const_0
              60: jle 10        iza then grane
drugi uslov   63: load_1
              64: const_2
              65: jge 5         iza then grane
then grana    68: const_2
              69: store_0
iza then      70: ...
grane
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+8 МН НМ УС KZ КБ ВТ КМ СМ РВ СЧ
Kristijan Ziza Ксенија Булатовић Војислав Томашевић Катарина Милошевић Сретен Мандић Радивоје Васиљевић Софија Чоловић

Zadatak 3.

```
if              70: load_0
                71: const_0
                72: jge ???      ← na then granu
drugi uslov     75: load_1
                76: const_2
                77: jgt ???      ← iza then grane
then grana      80: const_3
                81: store_0
iza then        82: ...
grane
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+8 МН НМ УС KZ Kristijan Ziza КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
if              70: load_0
                71: const_0
                72: jge 8          na then granu
drugi uslov     75: load_1
                76: const_2
                77: jgt 5          iza then grane
then grana      80: const_3
                81: store_0
iza then        82: ...
grane
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+8 МН НМ УС KZ Kristijan Ziza КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
if              82: load_0
                83: const_0
                84: jeq ???      ← na drugi deo ||
drugi uslov     87: load_1
                88: const_2
                89: jgt ???      ← na then granu
drugi deo ||    92: load_0
                93: const_1
                94: jne ???      ← iza then grane
then grana      97: const_4
                98: store_0
iza then        99: ...
grane
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+8 МН НМ УС KZ Kristijan Ziza КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 3.

```
if            82: load_0
              83: const_0
              84: jeq 8          na drugi deo ||
drugi uslov   87: load_1
              88: const_2
              89: jgt 8          na then granu
drugi deo ||  92: load_0
              93: const_1
              94: jne 5          iza then grane
then grana    97: const_4
              98: store_0
iza then      99: ...
grane
```

```
program ABC
{
    void main()
        int x, y;
    {
        if(x==1) y = 2;
        else y = 3;

        for(x=0; x<3; x++) print(x);

        while(x < 4){
            if(x==1) continue;
            else if(x==2) break;
            else y = 0;
            y = y % 2;
        }

        if(x>0 && y<2) x = 2;
        if(x>=0 || y<=2) x = 3;
        if(x!=0 && y>2 || x==1) x=4;
    }
}
```


+8 МН НМ УС KZ Kristijan Ziza КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 4.

Napisati kompletan MJ bajtkod koji odgovara datom programu.

Za svaku liniju izvornog koda naznačiti koji deo bajtkoda joj pripada.

```
program ABC
int x, y;
class TCalc{
        int tax;
        {
            void m(int i){ tax = i; }
            void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
        { void m(int i){ f(i); }}
    }
{
    int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+8 МН НМ УС KZ Kristijan Ziza КБ Ксенија Булатовић ВТ Војислав Томашевић КМ Катарина Милошевић СМ Сретен Мандић РВ Радивоје Васиљевић СЧ Софија Чоловић

Zadatak 4.

```
program ABC
int x, y;
class TCalc{
        int tax;
        {
    0   void m(int i){ tax = i; }
        void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
        { void m(int i){ f(i); }}
    }
{
    int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+7 РВ НБ КМ КМ KZ Kristijan Ziza АН Алекса Новаковић Стефан Теслић ТЂ Тијана Ђокић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић

Zadatak 4.

```
0: enter 2 2
3: load_0        ← this
4: load_1        ← i
5: putfield 1    ← nulto polje je pokazivač na TVF
8: exit
9: return
```

```
program ABC
int x, y;
class TCalc{
        int tax;
        {
    0   void m(int i){ tax = i; }
        void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
        { void m(int i){ f(i); }}
    }
{
    int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+5 МШ НБ КМ ФС KZ Kristijan Ziza АН Алекса Новаковић Стефан Теслић РВ Радивоје Васиљевић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић

Zadatak 4.

```
 0: enter 2 2
 3: load_0        ←  this
 4: load_1        ←  i
 5: putfield 1    ←  nulto polje je pokazivač na TVF
 8: exit
 9: return
10: enter 2 2
13: exit
14: return
```

```
program ABC
int x, y;
class TCalc{
        int tax;
        {
    0   void m(int i){ tax = i; }
   10   void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
        { void m(int i){ f(i); }}
    }
{
    int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+6 МШ НБ КМ ФС KZ Kristijan Ziza АН Алекса Новаковић Стефан Теслић РВ Радивоје Васиљевић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић

Zadatak 4.

```
 0: enter 2 2
 3: load_0          ←  this
 4: load_1          ←  i
 5: putfield 1      ←  nulto polje je pokazivač na TVF
 8: exit
 9: return
10: enter 2 2
13: exit
14: return
15: enter 2 2
18: load_0          ←  argument this
19: load_1
20: load_0
21: getfield 0      ←  pokazivač na TVF
24: invokevirtual 'f' -1
33: exit
34: return
```

```
program ABC
int x, y;
class TCalc{
        int tax;
        {
    0       void m(int i){ tax = i; }
   10       void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
{
    int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+5 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza

Zadatak 4.

Int fun nije clan neke klase,vec je globalna fja, pa nema skriveni argument vec samo ovaj jedan int arg

```
 0: enter 2 2
 3: load_0
 4: load_1
 5: putfield 1
 8: exit
 9: return
10: enter 2 2
13: exit
14: return
15: enter 2 2
18: load_0
19: load_1
20: load_0
21: getfield 0
24: invokevirtual 'f' -1
33: exit
34: return
```

```
35: enter 1 1
38: load_0
39: load_0
40: mul
41: exit
42: return
```

```
program ABC
int x, y;
class TCalc{
        int tax;
        {
    0   void m(int i){ tax = i; }
   10   void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
{
   35   int fun(int arg){
            return arg * arg;
        }
        void main() TCalc obj;
        {
            x = fun(3);
            obj = new ATCalc;
            obj.m(4);
        }
}
```


+5 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza

Zadatak 4.

```
43: const 109      ← 'm'
48: putstatic 2
51: const_m1       ← -1 (kraj naziva metode)
52: putstatic 3
55: const 0        ← adr(TCalc::m)
60: putstatic 4
63: const 102      ← 'f'
68: putstatic 5
71: const_m1       ← -1
72: putstatic 6
75: const 10       ← adr(TCalc::f)
80: putstatic 7
83: const -2       ← -2 (kraj TVF za TCalc)
88: putstatic 8
```

```
program ABC
int x, y;
class TCalc{   TVF = 2
        int tax;
        {
    0   void m(int i){ tax = i; }
   10   void f(int i){ }
        }
    }
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
{
   35   int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+7 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza

Zadatak 4.

```
43: const 109        91: const 109        <- 'm'
48: putstatic 2      96: putstatic 9
51: const_m1         99: const_m1         <- -1
52: putstatic 3     100: putstatic 10
55: const 0         103: const 15         <- adr(ATCalc::m)
60: putstatic 4     108: putstatic 11
63: const 102       111: const 102        <- 'f'
68: putstatic 5     116: putstatic 12
71: const_m1        119: const_m1         <- -1
72: putstatic 6     120: putstatic 13
75: const 10        123: const 10         <- adr(ATCalc::f)
80: putstatic 7     128: putstatic 14
83: const -2        131: const -2         <- -2
88: putstatic 8     136: putstatic 15
```

```
program ABC
int x, y;
class TCalc{                    TVF = 2
        int tax;
        {
    0       void m(int i){ tax = i; }
   10       void f(int i){ }
        }
    }                           TVF = 9
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
 {
   35 int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+7 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza

Zadatak 4.

```
139: enter 0 1
142: const_3
143: call -108        ← 35 - 143
146: putstatic 0
```

```
program ABC
int x, y;
class TCalc{        TVF = 2
        int tax;
        {
    0   void m(int i){ tax = i; }
   10   void f(int i){ }
        }
    }               TVF = 9
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
{
   35   int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+7 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza

Zadatak 4.

```
139: enter 0 1
142: const_3
143: call -108
146: putstatic 0
149: new 8          ← polje tax + TVF
152: store_0
153: load_0
154: const 9        ← adresa TVF se smešta
159: putfield 0     ← u nulto polje
```

```
program ABC
int x, y;
class TCalc{        TVF = 2
        int tax;
        {
    0   void m(int i){ tax = i; }
   10   void f(int i){ }
        }
    }               TVF = 9
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
{
   35   int fun(int arg){
        return arg * arg;
    }
    void main() TCalc obj;
    {
        x = fun(3);
        obj = new ATCalc;
        obj.m(4);
    }
}
```


+7 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza

Zadatak 4.

```
139: enter 0 1
142: const_3
143: call -108
146: putstatic 0
149: new 8
152: store_0
153: load_0
154: const 9
159: putfield 0
162: load_0          ← argument this
163: const_4
164: load_0
165: getfield 0      ← pokazivač na TVF
168: invokevirtual 'm' -1
177: exit
178: return
```

```
program ABC
int x, y;
class TCalc{   TVF = 2
        int tax;
        {
    0   void m(int i){ tax = i; }
   10   void f(int i){ }
        }
    }          TVF = 9
    class ATCalc extends TCalc{
   15   { void m(int i){ f(i); }}
    }
{
   35   int fun(int arg){
            return arg * arg;
        }
        void main() TCalc obj;
        {
            x = fun(3);
            obj = new ATCalc;
            obj.m(4);
        }
}
```


+7 КМ ФС НБ РВ АН Алекса Новаковић Стефан Теслић УС Урош Стоковић МН Милош Настић КБ Ксенија Булатовић ВТ Војислав Томашевић KZ Kristijan Ziza